



National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

THE UNIVERSITY OF ALBERTA
MONITORING SYSTEM AND CALIBRATION
OF A
LIQUID HYDROGEN POLARIMETER

BY
RYUICHI IGARASHI

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND
RESEARCH IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE

IN
NUCLEAR PHYSICS

DEPARTMENT OF PHYSICS

EDMONTON, ALBERTA

FALL, 1987

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-40914-2

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: RYUICHI IGARASHI

TITLE OF THESIS: MONITORING SYSTEM AND CALIBRATION OF
A LIQUID HYDROGEN POLARIMETER

DEGREE: MASTER OF SCIENCE

YEAR THIS DEGREE GRANTED: 1987

Permission is hereby granted to THE UNIVERSITY OF
ALBERTA LIBRARY to reproduce single copies of this thesis and to lend
or sell such copies for private, scholarly or scientific research purposes
only.

The author reserves other publication rights, and neither the thesis
nor extensive extracts from it may be printed or otherwise reproduced
without the author's written permission.

Ryuchi Igarashi.....

Permanent Address: 8005-154 Street

Edmonton, Alberta

T5R 1S2

Date: *5 Aug. 1987*.....

THE UNIVERSITY OF ALBERTA
FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled MONITOR SYSEM AND CALIBRATION OF A LIQUID HYDROGEN POLARIMETER submitted by Ryuichi IGARASHI in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE.

Gerald Ray
.....
(Supervisor)

J. C. Lumsden
.....
J. Ray
.....
G. Lumsden
.....

Date: *10 July 1987*
.....

To my mother
Yuriko Kitamura-Igarashi,

who, like all mothers, has to put up
with a lot of nonsense from her children,
but still cares a great deal for them.

Abstract

In an attempt to understand short range nucleon-nucleon interactions, a polarimeter has been constructed to measure deuteron tensor polarizations from electron-deuteron scattering. The polarimeter uses deuteron-proton elastic scattering in liquid hydrogen as the analyzing interaction. This thesis describes the construction of the liquid hydrogen target flask for the polarimeter, the design of the computer monitor system for the liquid hydrogen target system, and a vector polarization calibration analysis after calibration experiments at the Laboratoire Nationale Saturne, Saclay, France, using this polarimeter.

Acknowledgments

Special thanks to Dr. Gerald Roy and to the late Dr. Douglas Sheppard for guidance with the monitor system, in the physics, and in the preparation of my thesis.

Many thanks go to Dr. John M. Cameron who often gave me some of his time to check my progress and give guidance, to Jan Soukup who supervised the construction of the target flask, to James Pasos who supervised the data analysis programming, to Dr. William Zeigler, Dr. Harry Fielding, and Dr. Nathan Rodning with whom I consulted for the physics and the experimental aspects of the data analysis, and to Dr. Larry Antonuk who, along with Dr. Roy and Dr. Cameron, was very helpful with getting things done at Saclay.

-- The technical help from Wayne Malkin with programming the Apple II, from Randy Dorosh with the hardware aspect of the monitor, from Earnst Pierce with the target flask, from Earl Cairns, and from Lars Holme is also very much appreciated. For the clerical assistance and for the assistance on the finer points of bureaucratic paperwork, I have Greta Tratt and Llanca Letelier to thank.

Thanks also go to Warren Anderson, Mark Bourassa, Bill Miner, Ado Umezawa, Larry Yakiwczuk, Vahe Ghazikhanian, Remy Dawson, Jon Johansen, Phillip Langill, Gerhard Lotz, Paul McNeely, and Khaled Negm, fellow students with whom I had many discussions about many aspects of physics (and other topics), and aided me with my work and my studies.

And of course, thanks go to my family and family friends who were my foundation of support throughout my undergraduate and Master's studies.

TABLE OF CONTENTS

CHAPTER

Page

I. Introduction.	1
References for Chapter I.	11
II. The AHEAD Target Flask.	12
1. The Front Window.	14
2. The Rear Window.	15
3. The Cylinder.	15
4. Final Assembly.	18
III. The Monitor System.	20
1. Requirements.	20
2. The Two Computers.	22
3. Program Details.	27
a. SAFE.	29
b. Apple II.	38
i. Screens.	40
ii. Processing.	45
iii. Commands.	49
iv. Interrupts.	52
c. Start-up Program.	55
d. Support Software.	58
References for Chapter III.	63

IV. Asymmetry Analysis	64
1. Theory.	64
2. The Analy.	69
3. Program Details.....	80
References for Chapter IV.....	88
V. Target and Monitor Performance	89
1. Target Cryogenic Performance.	89
2. Possible Improvements on the Monitor	93
References for Chapter V	98
Bibliography	99
Appendix A: Polarization and Cross Section.....	101
References for Appendix A	104
Appendix B: Wire-E Counter Correspondence	105
Appendix C: The Least Squares Algorithm.	112
Appendix D: Monitor Instructions.	115
Appendix E: Monitor Program Listings	120
E.1: H2 SAFE	120
E.2: H2 PC	130
E.3: H2 PC BOOT	142
E.4: H2 PC SET.....	142
E.5: Data Transfer Routine	145

LIST OF TABLES

Table	Description	Page
III-1	Location of Data Points on SAFE 8000 I/O Cards.	60
III-2	Location of Control Points on Output Card	61
III-3	List of Data Points by Priority Interrupt.	61
III-4	Graph Screen Graduations	62
IV-1	Asymmetries for different Solenoid Currents and different analyses.	84
IV-2	Null Asymmetry Solenoid Currents and Parameters from sine fits from different analyses.	84

LIST OF FIGURES

Figure	Description	Page
I-1	Various Theoretical Calculations of t_{20} vs. Momentum Transfer (q)	3
I-2a	iT_{11} vs. C.M. Angle for d-p elastic scattering at 191 MeV	5
I-2b	iT_{20} vs. C.M. Angle for d-p elastic scattering at 191 MeV	5
I-2c	iT_{22} vs. C.M. Angle for d-p elastic scattering at 191 MeV	5
I-3	Schematic Design of the AHEAD Polarimeter.	6
I-4	Wire Wraparound in Outer and Inner Wire Chambers.	8
I-5	E Counter Array for Vector Polarization Runs	8
II-1	AHEAD Target Flask and Components.	13
II-2	Jig for Pressing Front and Rear Windows.	13
II-3	Schematic Side View of Sanding Jig.	17
III-1	Schematic Representation of the AHEAD Target System and data points.	21
III-2	Schematic of Monitor System Hardware.	23
III-3	Memory Map of Apple IIe	26
III-4	Schematic of Monitor Program Logic.	28
III-5a	Logic of Safe 8000 Monitor Point Scan	30
III-5b	Logic of Safe 8000 Timers Scan.	30
III-5c	Logic of Safe 8000 Interrupt Routine.	30
III-5d	Logic of Communication Routines for Data Transfer	30
III-6	Diagnostic Logic for Target Cell Alarms.	36
III-7	Representation of Monitor Screens.	41-43
III-8	Log(pressure) vs. Gauge Voltage for vacuum gauges.	48
IV-1	Coordinate Systems for Polarizations	66

IV-2	iT_{11} vs. Lab Angle for d-p elastic scattering at 191 MeV.....	70
IV-3a	Raw Event Breakdown, Q-cuts of 0-15 cm	71
IV-3b	Bad Event Breakdown of Triple Coincidence Events.....	71
IV-4	Kinetic Energy vs. Lab Angle of <u>Elastically</u> Scattered 170 MeV Deuteron.	73
IV-5	Lab Angle Envelope for Q-cuts	74
IV-6	Raw ΔE vs. E plot for $\Delta E6$ and E15.....	75
IV-7	Raw Time-of-Flight vs. E for E15.....	77
IV-8a	ΔE vs. E for E15 passing all cuts.....	78
IV-8b	ΔE vs. E for E15 failing any cut.....	79
IV-9a	On-line Asymmetry vs. Solenoid Current for d-p elastic scattering at 170 MeV.....	85
IV-9b	Asymmetry vs. Solenoid Current for d-p elastic scattering at 170 MeV, Q-cut of 0-15 cm	86
IV-9c	Asymmetry vs. Solenoid Current for d-p elastic scattering at 170 MeV, Q-cut of 15-30 cm.	87
V-1a	Target Sensors during Preliquefaction Cooldown.....	90
V-1b	Target Sensors during Full Cooldown	90
V-2	Log(pressure) vs. Gauge Voltage with polynomial fits for vacuum gauges	96
B-1	Geometry for Mathematical Calculations of Wire Correspondences.....	107
B-2a	Wire Correspondences on Right Side with 2.5 cm radius beamspot.....	108
B-2b	Wire Correspondences on Right Side with 5 cm radius beamspot.....	109
B-3	Wire Correspondences on a center sector.....	111

Chapter One. Introduction

The work for this thesis centered on the Alberta High Efficiency Analyzer for Deuterons (AHEAD) polarimeter. This device was constructed to measure tensor polarizations of recoil deuterons from an electron-deuteron reaction in an experiment to be carried out in the summer of 1987 at the Bates Linear Accelerator Center in Boston. This experiment is a large international collaboration involving scientists from the University of Alberta; the Bates Linear Accelerator Center, Massachusetts, USA; the Laboratoire Nationale Saturne, Saclay, France; and Syracuse University, New York. The work described in this thesis concentrates on the construction and calibration of AHEAD, not the actual experiment.

The purpose of the Bates experiment is to measure tensor polarizations of deuterons, which can shed light on the effects of various contributions in electron-nucleon and nucleon-nucleon interactions (the deuteron is of particular interest in this because it is the only relatively stable 2-nucleon system known). The differential cross section for unpolarized electrons scattering off unpolarized deuterons is

$$\sigma = \sigma_0 \left[F_C^2 + \frac{2}{3} b F_M^2 + \frac{8}{9} b^2 F_Q^2 + \frac{4}{3} b(1+b) F_M^2 \tan^2\left(\frac{\theta}{2}\right) \right] \quad (\text{I-1})$$

$$\text{where } b = q^2 / 4M_d^2$$

q is the momentum transfer in the interaction, and F_M , F_C , F_Q are the magnetic, electric monopole, and electric quadrupole form factors for the deuteron. These quantities represent their respective source distributions, and are essentially Fourier transforms of their associated potentials. Here, F_M is easily determined by angular distribution of recoiling deuterons, but F_C and F_Q cannot be separated. Measurements of polarizations t_{20} , t_{21} and t_{22} could solve this problem since,

$$\sigma_0 t_{22} = \frac{-1}{2\sqrt{3}} F_M^2 \quad (\text{I-2a})$$

$$\sigma_0 t_{21} = \frac{4}{\sqrt{3}} [b + b^2 \sin^2(\frac{\theta}{2})]^{1/2} F_M F_Q \sec(\frac{\theta}{2}) \quad (I-2b)$$

and

$$\sigma_0 t_{20} = \frac{4}{3\sqrt{2}} \{ 2b F_C F_Q + \frac{2}{3} b^2 F_Q^2 + b[1 + 2(1+b)] \tan^2(\frac{\theta}{2}) F_M^2 \} \quad (I-2c)$$

This system can be solved to yield all three form factors. The primary concern is for t_{20} for which theories are uncertain. Some contributions in question are meson exchange currents, relativistic corrections, and six quark contributions (see figure I-1).¹

AHEAD is to be used to measure these polarizations, based on deuteron-proton (d-p) elastic scattering. The cross section, or equivalently the number of counts, in the scattering frame for this reaction is ^{2,3}

$$\begin{aligned} \sigma(\theta) = \sigma_0(\theta) [& 1 + 2\text{Re}(it_{11})\text{Re}(iT_{11}(\theta)) + t_{20}T_{20}(\theta) \\ & + 2\text{Re}(t_{21})\text{Re}(T_{21}(\theta)) + 2\text{Re}(t_{22})\text{Re}(T_{22}(\theta))] \end{aligned} \quad (I-3)$$

where $\text{Re}(it_{11})$ is the vector polarization of the incident deuteron, and t_{20} , $\text{Re}(t_{21})$, and $\text{Re}(t_{22})$ are the tensor polarizations, and the T_{ij} are their respective analyzing powers (which contain all the information about the scattering reaction). This is related to the incident frame by ⁴

$$\text{Re}(it_{11}) = \text{Re}(it_{11}' e^{-i\phi}) \quad (I-4a)$$

$$t_{20} = t_{20}' \quad (I-4b)$$

$$\text{Re}(t_{21}) = \text{Re}(t_{21}' e^{-i\phi}) \quad (I-4c)$$

$$\text{Re}(t_{22}) = \text{Re}(t_{22}' e^{-2i\phi}) \quad (I-4d)$$

where ϕ is the azimuthal angle the scattering plane makes with the incident x' - z' plane. By taking counts at various angles of ϕ , it is sometimes possible to use equations I-3 and I-4 to solve for the t_{ij}' , which are the observables desired. Generally, these solutions are manifestations of asymmetries in the azimuthal distribution of counts. For the Bates experiment, $\text{Re}(it_{11})$ will not be produced, hence AHEAD will not measure its contribution. But $\text{Re}(it_{11})$ was necessary for calibration, which is discussed later.

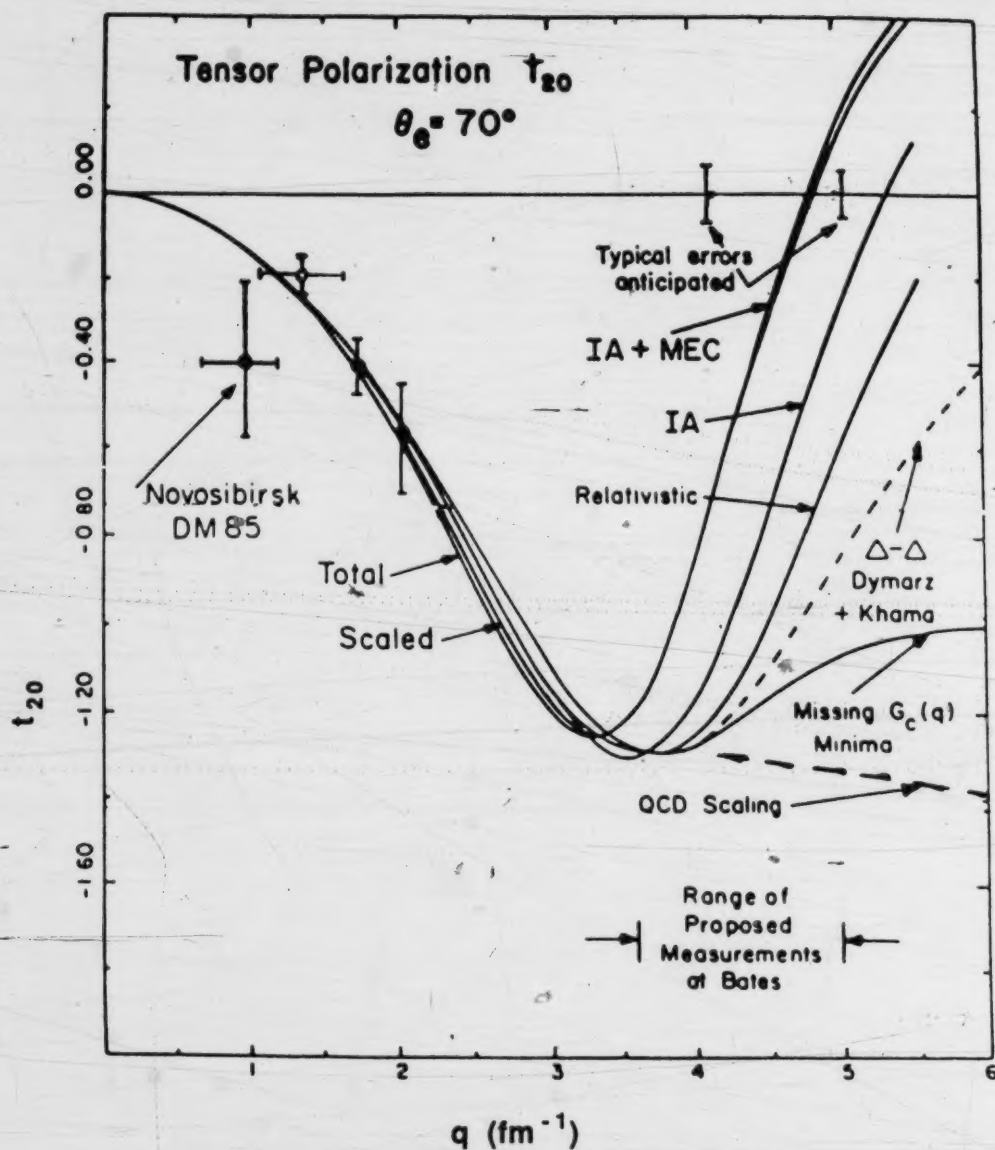


Figure I-1: Various Theoretical Calculations¹ of t_{20} vs. momentum transfer (q). IA is Impulse Approximation, MEC is Meson Exchange Currents. Note that the proposed region of investigation at Bates has diverse predictions.

D-p elastic scattering was chosen for the magnitude (see figure I-2) and small energy dependence of its analyzing powers. Equation I-3 indicates that the count rate at a given point in space can be affected by the magnitude of the polarization. But if the analyzing power is insignificant, there would be no way to observe these polarization effects. Energy dependences introduce sensitivity to experimental method. For example, the target in AHEAD is over 25 cm long, subjecting the incoming beam to energy degradation. Then the analyzing power for each event may vary substantially, making computations difficult.

In order to reliably measure these quantities, a polarimeter was needed that could make observations at all azimuthal angles (ϕ). Hence, AHEAD took on a cylindrical geometry. This is also part of a precaution against artificial asymmetries - by changing or rotating the incident polarization and by correlating measurements, the detectors can be checked against each other for consistency. The polarimeter (see figure I-3) consisted of a liquid hydrogen (LH_2) target and support system, 2 thin plastic scintillators in front of the target for coincidence triggering and timing, 2 x-y wire chambers in front of the target, 2 concentric cylindrical wire chambers (self quenching streamer chambers with 60 cells in the inner chamber and 120 cells in the outer), a hexagonal array of 6 thin plastic energy loss scintillators, and an array of 18 total energy calorimetry plastic scintillators. The front end wire chambers tracked incoming deuterons and the cylindrical chambers tracked outgoing particles. Particle identification was by energy loss (ΔE), and by time-of-flight (TOF) from the front end chambers to the ΔE counter, both with respect to total energy (E). Data acquisition was performed by a J11 STARBURST microprocessor through CAMAC and FASTBUS. Data was then transmitted to a DEC LSI 11 which stored it on high density tape.

There were at least two peculiarities in AHEAD that complicated interpretation of data. A minor one was that since the photomultiplier outputs and the wire chamber outputs came out of opposite ends of AHEAD, the numbering schemes for the two types

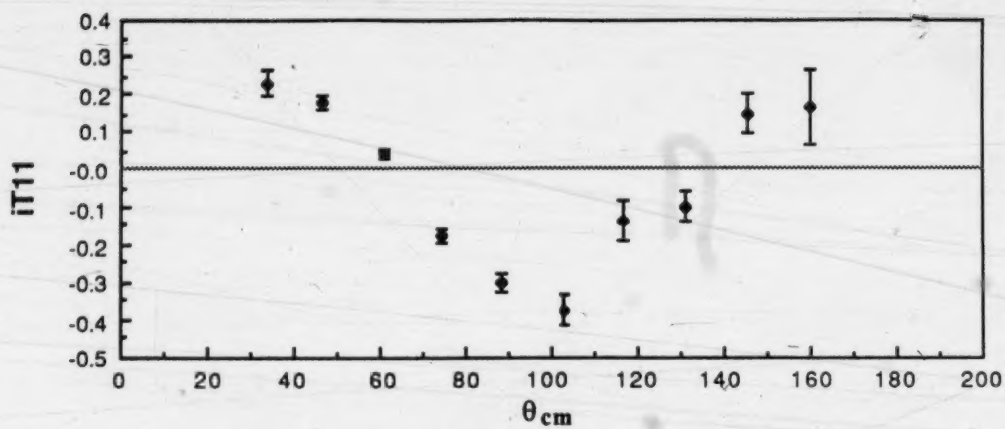


Figure I-2a: iT_{11} vs. C.M. Angle for d-p elastic scattering at 191 MeV.⁵

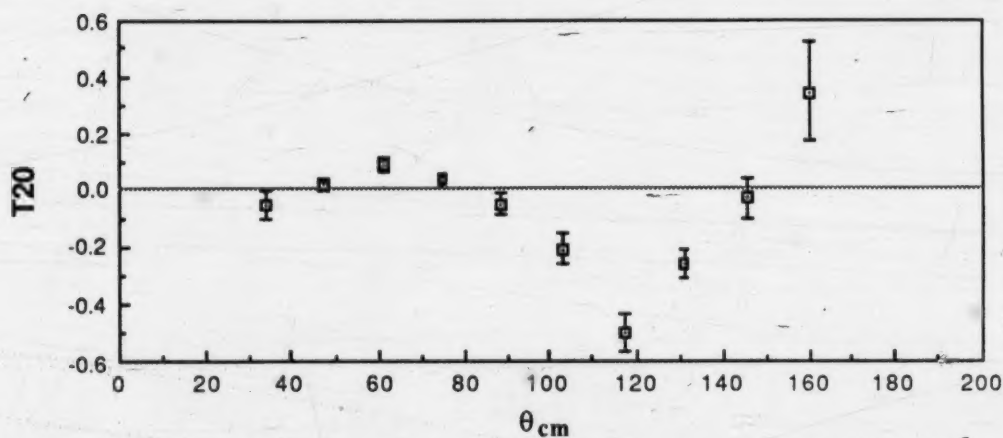


Figure I-2b: T_{20} vs. C.M. Angle for d-p elastic scattering at 191 MeV.⁵

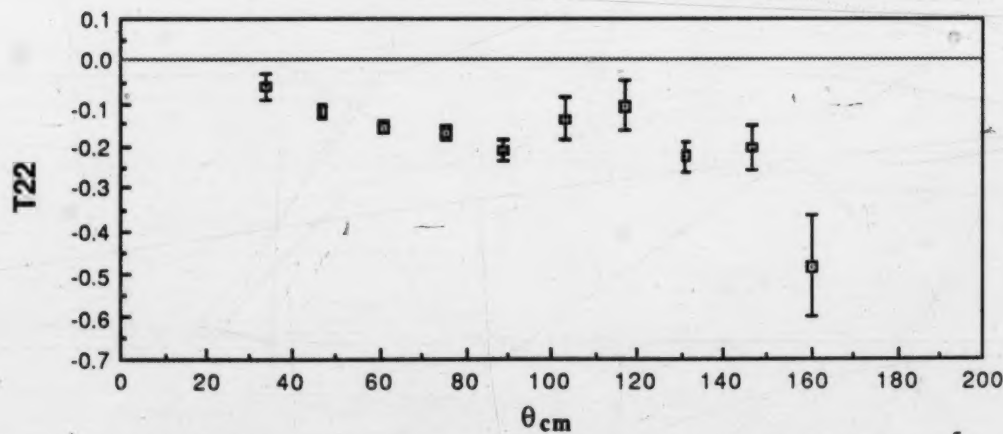


Figure I-2c: T_{22} vs. C.M. Angle for d-p elastic scattering at 191 MeV.⁵

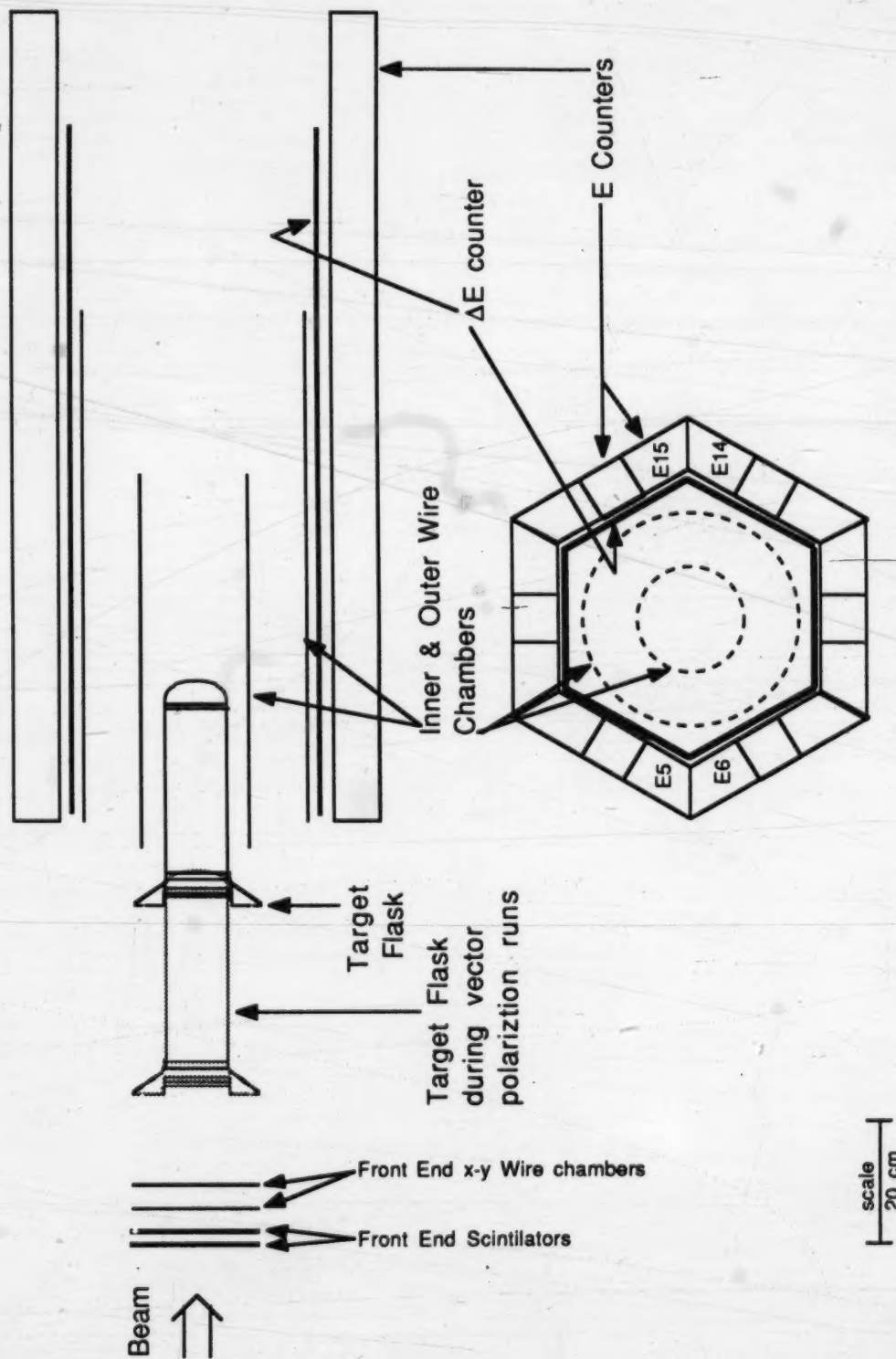


Figure I-3: Schematic Design of the AHEAD Polarimeter

of detectors ran counter to each other. Looking down the beam, the wire chambers were numbered clockwise and the E and ΔE counters were numbered counter clockwise, with #1 starting at the top.

The second peculiarity was in the wire chamber design. Instead of ending each wire at the opposite end of the chamber, it wrapped around to a cell three positions down and continued back to the front of the detector (see figure I-4). Only three adjacent wires can wrap around this way, so all the wires in the detector are grouped in threes (six wire ends). This economized electronics but somewhat complicated interpretation. A distinction between wires and wire ends must always be made. Wire ends are associated with physical geometry. The wires are significant in software and analysis. A hit on one half of a wire can be attributed to the nearest wire end, and this gives azimuthal information. But a hit on one half will affect the other half, as the signal moves to both ends of the wire. So a hit will be registered at a wire end three positions down. This situation should be kept in mind when looking at wire data, though it is simple to decide which end the hit should be attributed to since the signal attenuates with distance traveled. And there are times where both concepts merge (see Chap IV and Appendix B).

In order to be an effective detector, it was necessary to calibrate AHEAD with incident deuterons of known polarization. This experiment was carried out in the summer of 1986 at the Laboratoire National Saturne (LNS). ΔE and E counter light pathlength attenuation calibrations were done during parasitic runs in the Radiography Hall in June 1986. During the first week of August, 170 MeV vector polarized beams were used to calibrate a spin precessing solenoid. And the remaining time, to the end of that month, was spent on the actual tensor calibration runs at several different incident deuteron energies. All the August runs were performed in the SPES I spectrometer hall. In order to calibrate the solenoid and to observe the progress of the experiment, the LSI also dumped some data to LNS's resident data acquisition computer system, SAR (Satellite d'Acquisition Rapide).

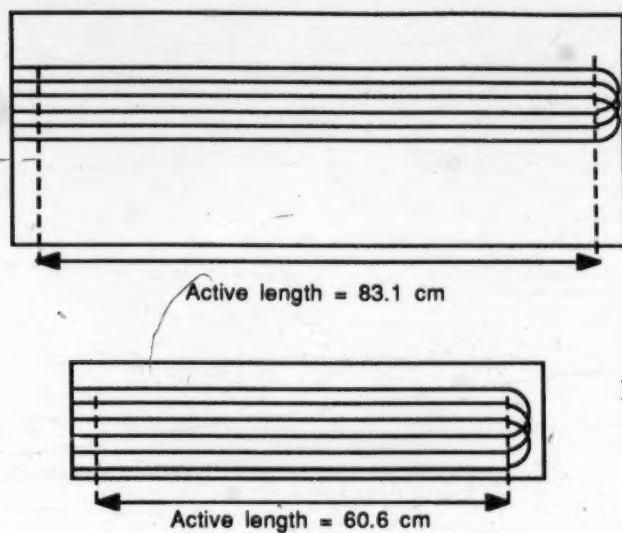


Figure I-4: Wire Wrap-around in Outer and Inner Wire Chambers

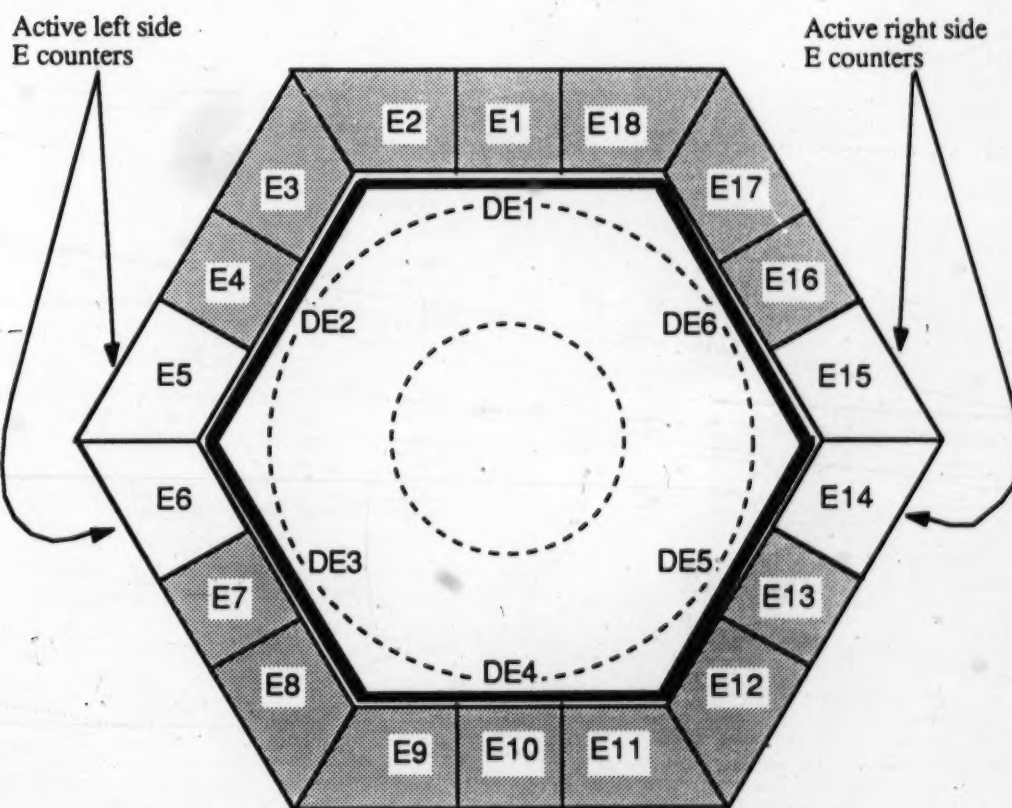


Figure I-5: E Counter Array for Vector Polarization Runs

The June run was to empirically ascertain the attenuation of light as it travelled and reflected through the scintillator on the way to the photomultipliers. The further down the scintillator and the more reflections, the more the light signals are absorbed. This calibration was done simply by orienting the detector array perpendicular to the beam, and lowering or raising the detectors so most of the beam passed through one E counter (and the E counter horizontally on the other side of the array) at a time. After a half hour, there were enough counts to make reliable calculations.

The vector polarization calibration, as will be detailed in Chapter IV, was necessary to find a solenoid current that would rotate the beam polarization by $\pi/2$. This rotation, in conjunction with a spin precessing dipole magnet, was necessary to create tensor polarizations that the accelerator could not produce. To find this current, the current was stepped from 0 to 240 A in approximately 50 A steps, keeping the beam energy at 170 MeV. Only E counters E5, E6, E14, and E15 were in the trigger (figure I-5) because vector polarization should only give left-right asymmetry. Each step took one to two hours, to acquire sufficient statistics. On-line analysis programs calculated asymmetries for each step. A rough fit then indicated a solenoid current at which the vector asymmetry was zero.

The tensor polarization runs involved taking data for several different energies. Saturne produced beams with beam pulses of four different polarization states for these runs. This was again for instrumental consistency checks. The tensor calibrations will not be discussed in this thesis.

Specifically, there were three projects for this thesis: 1) construction of the LH_2 container, or flask; 2) programming a computer monitor system for the LH_2 target system; 3) more rigorous vector polarization analysis. Chapter II explains how a LH_2 target cell is constructed. This is of some interest considering the materials used (plastic and steel), the temperatures and pressures it is subjected to (20 - 300 K and 1.1 atmosphere absolute), and the material contained (H_2 expands 788 times in volume from

liquid to gaseous state). Chapter III details the computer monitor system, particularly its programming and faults. This chapter is meant to be a reference for the system and will be fairly technical. Chapter IV explains the need for and the procedure of a vector polarization analysis. This analysis actually investigated asymmetry and did not involve computing polarizations (or analyzing powers). Chapter V is a summary of the target and monitor performance, and some suggestions on modifications for the monitor.

References

¹J. M. Cameron, M. E. Schulze, and W. Turchinets, Bates Updated Proposal #84-17, 1986.

²W. Haeberli, "Polarization Experiments" in Nuclear Spectroscopy and Reactions, Part A, ed. J. Cerny (New York: Academic Press, 1974), p. 159.

³See also Appendix A.

⁴Haeberli, p. 161

⁵M. Garcon, et al., "Measurement of Vector and Tensor Analyzing Powers for 191 and 395 MeV Deuteron Scattering," Nuclear Physics, A458 (1986), 294.

Chapter II. The AHEAD Target Flask

The AHEAD polarimeter used a mylar target flask mounted on a stainless steel base to contain the liquid hydrogen (LH_2). Its design, choice of materials, and tooling were completed before the author's involvement (that being the assembly of the flask).

The plastic component actually consisted of three parts: a flat front window, a 10 cm diameter cylinder, and a toruspherical end dome (see figure II-1). The front and end windows were made by sandwiching a mylar sheet between two ringlike plates with holes to accommodate a plunger, heating the sandwich, and pushing a plunger through, the plunger having the shape of the desired flask piece (see figure II-2). Though simple in principle, producing viable windows proved to be complicated, consuming three months to make three sets (i.e. 3 flasks). The cylinder was made by wrapping a sheet around a cylindrical jig and gluing the seams.

The end dome fitted onto the cylinder so it had a 10 cm diameter. The toruspherical shape is a rotated curve composed of two circular arcs: the face has a radius of curvature equal to the diameter of the cylinder (10 cm) and the edge has a radius of curvature of 1/4 diameter of the cylinder (2.5 cm). This shape is ideal for the endcaps of pressure vessels. The actual depth of the enddome piece was 4.3 cm, including lap joint.

The front window was not connected to the cylinder but to the steel work. It had a diameter of 10.7 cm and depth of 1.4 cm. The edges were rounded to accommodate some bulging due to internal flask pressure. A retaining ring was glued over the side of the window to keep a tight seal. The same was done to the inside of the cylinder at the lap joint (figure II-2).

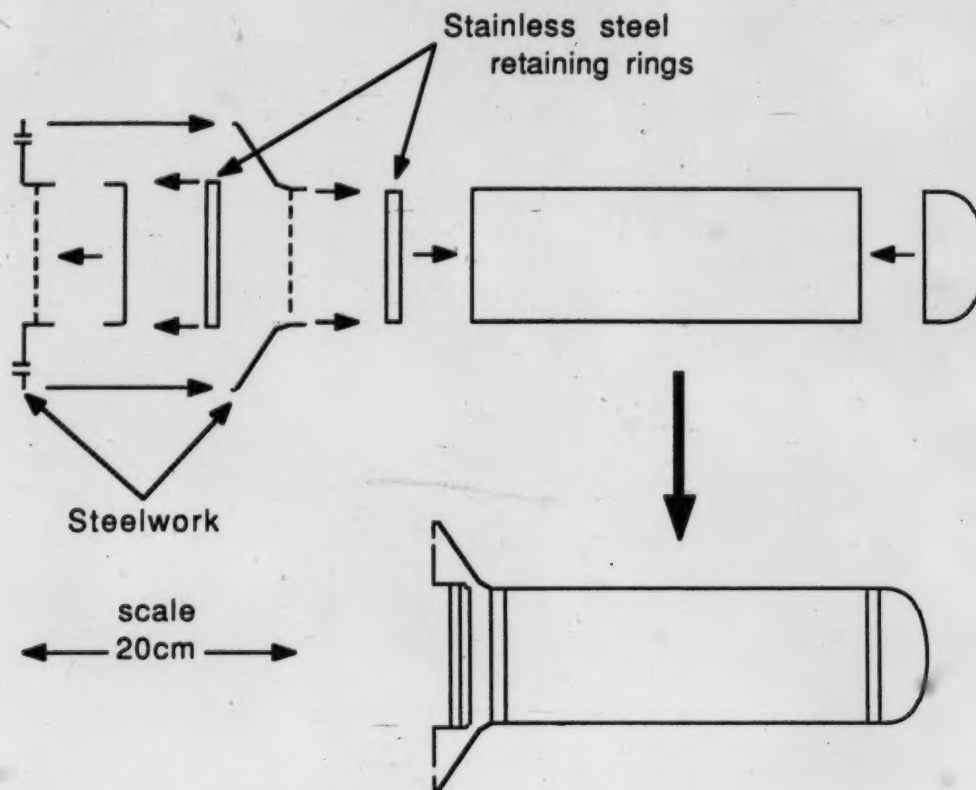


Figure II-1: AHEAD Target Flask and Components

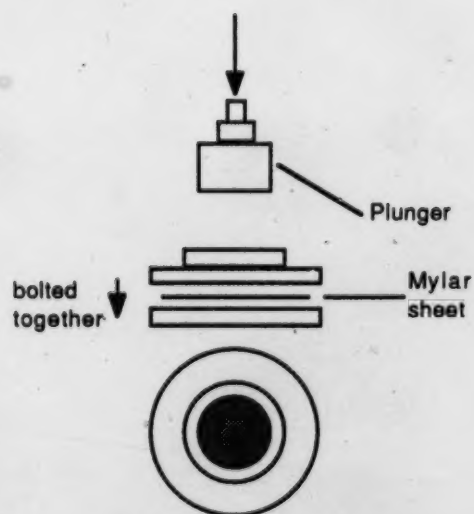


Figure II-2: Jig for Pressing Front and Back Windows, side and top view.

1. The Front Window

Circular sheets were cut out of a roll of 0.25 mm mylar, approximately 21.5 cm in diameter. This was just slightly smaller than that required to keep clear of the holes for the clamping bolts on the sandwiching plates. This way the sheet could be roughly centered by sighting down the bolt holes for protruding mylar. While the oven was preheating to 140°C, the sheet was given a methanol wash, without rubbing. This was to remove dirt and dust, attached by static, that would gouge the mylar when the plunger is pushed through. The rings and plunger were made of anodized aluminium, and were also cleaned. Before the mylar was placed in the rings, the lip of the lower ring was lubricated with a thin film of silicone lubricant. The lubricant was applied then wiped off to limit the thickness of the film. If care is not taken with the lubricant, it tends to spread between the mylar and the jig surface, allowing too much slippage and causing wrinkles.

Once the mylar is placed between the plates and centered, the jig must be tightened. This proved to be the major stumbling block. Too tight and the mylar snapped during pressing. Too loose and not enough mylar was held back to allow uniform stretching, causing wrinkles. Not uniform enough and stretching was uneven, again causing wrinkles.

The whole assembly was heated in the oven for 1-2 hours. At 140°C, the mylar was soft enough to stretch without experiencing undue stress, but neither was it fluid.

The next step had to be done as quickly as possible to prevent loss of malleability during pressing. The plunger was mounted on a press. The jig was centered roughly under the plunger, then the plunger was eased into the jig for final centering, and was then pressed about 1.5 cm. The press was locked in this position for approximately one hour while the whole assembly slowly and uniformly cooled.

After cooling, the plastic and plunger usually jammed in the jig. The only solution was to remove the top plate, cut off the excess collar mylar as close to the plunger as

possible (without scratching the plunger). The plunger with plastic was then pushed through the bottom plate.

2. The Rear Window

The rear domes were more difficult to deal with. The primary problem was that the dome, having a greater surface area, required more plastic to be pulled. This tended to aggravate the kind of problems encountered with the Front window.

Another difference, which caused some inconvenience, was that the jig for the rear window was made of brass. This jig was loaned to the AHEAD group by TRIUMF, and was used because the flask was based on dimensions of another target flask that already existed. Brass is substantially heavier than aluminium and takes longer to heat and cool. It also required frequent polishing. Heat transfer was the most notable difference (though the weight was also a substantial nuisance). The jig had to be heated for at least two hours, and cooled two hours just to be hot to touch (about 50°C).

One advantage of the TRIUMF jig was that the plunger had cooling tubes bored into it. This facility was not used immediately after pressing because the mylar tended to shrink with the plunger, making the piece too small, and to wrinkle due to inhomogeneity. But at the removal stage, blowing liquid nitrogen (N_2) through released the plunger (with the dome, if the top ring was removed) from the jig. This reduced the risk of trauma to the dome.

3. The Cylinder

Rectangular sheets, 22.5cm x 32.4cm, were cut out of a roll of mylar. These sheets were cut so that the curl of the tube went with the curl of the roll (i.e. 22.5cm across the roll, 32.4cm into) to eliminate unnecessary stress. Great care was taken to cut

as perfectly rectangular sheets as possible, to maintain cylindrical symmetry during the actual experiment. A scalpel was used to etch lines to cut. One side was etched and cut at a time, so that corrections, if possible, could be made to maintain symmetry at the next cut.

—Since the target was constructed of various parts glued together, the glue joints had to be sandblasted to provide adhesive surfaces. Ordinary sanding would have introduced cuts which were potential rupture points (despite this, sanding was used for tapering, later). After some experimentation, it was found that sandblasting at a 45° angle to the surface with coarse sand at low pressure provided the most satisfactory results. Fine sand was tried at first but it tended to embed into the plastic in copious amounts, and to dimple the reverse side. So all plastic parts were masked with masking tape and a 1cm glue joint sandblasted at the Physics Machine Shop.

Another effort to maintain symmetry of the target was to taper the overlap joint of the cylinder, to try to maintain an even thickness of plastic throughout the cylinder. This was done by hand, due to shortage of manpower at the shops, using a jig which tilted a sanding bar at the correct angle (see figure II-3). Medium, then fine grit emery paper was used, and took 30-40 minutes for the two edges. The result was not an even taper, but it did reduce the thickness of plastic at the joint substantially.

The tapered edges of the sheet were glued together to form the necessary cylinder, using an epoxy glue: a 1:1 mixture of Versamid 140 and Epon 828 (or equivalent) and a small drop (literally) of Silane. To avoid excessive amounts of glue, it was applied, then wiped along the edge once, and then across the edge, the latter to provide a path for trapped air bubbles to escape during clamping and setting. The remaining thin film of glue was usually sufficient to form a completely sealed joint.

To form a cylinder of correct dimensions, the sheet was wrapped around an anodized solid aluminium cylinder and clamped at the joint. This mandrel was cut to match the inside diameter of the flask to be. To avoid sticking to the cylinder, various

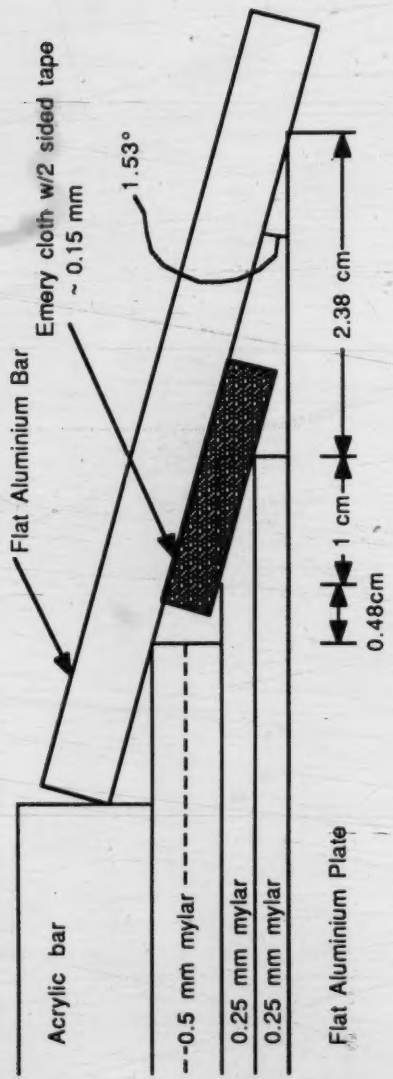


Figure II-3: Schematic side view of Sanding Jig for tapering longitudinal lap joint of cylinder.

anti-stick substances were tried, eventually settling on a soft wax buffed onto the mandrel.

It was found that simply clamping at the joint while the glue set was not sufficient. The mylar had a tendency to shift and twist, leaving a misaligned joint or misshapen cylinder. So to keep the mylar firmly and tightly on the mandrel, a strip of polyethylene was placed over the joint, side by side, down the joint. Another strip of polyethylene was placed over this, as a cushion against the lucite and steel bar clamp.

The glue took approximately 24 hours to set, after which the clamp and tape were removed. Despite the care to avoid sticking with the clamping pressure involved, enough anti-stick wax was squeezed out to cause sticking. It was easily eliminated by blowing compressed air underneath the mylar, but not directly at the joint (to avoid damage there). After this, the cylinder simply slid off.

4. Final Assembly

All the plastic parts were glued with the same epoxy mix as the cylinder joint. In order for the glue at the rear dome to be as thin as possible and to eliminate bubbles that cause leaks, it was necessary to clamp the joint. This was done by an expandable teflon ring inserted inside the cylinder, at the joint, and a PVC ring outside the dome, at the joint. The teflon ring was expanded by screwing two tapered caps together.

There remained the matter of whether or not the flask could really withstand the pressures and temperatures to be encountered during the experiment. During normal running, the internal pressure of the flask was to vary between 16 to 17 psia. Also, it was predicted that pressures as high as 2 atmospheres could be encountered during filling or emptying. A jig that used the flask as a bell jar was made. Four tests using bottled helium gas were performed. Three flasks were destroyed because they couldn't be clamped down hard enough to prevent them from blasting themselves upwards, like

rockets. It was found that the flasks were capable of withstanding over 2.5 atmospheres at room temperature. The pressure tests were positive and the final assembly was carried out.

The front window was glued onto one end of steelwork (which carried the H_2 feed tubes and level sensors), and the cylinder (with dome) at the other. The glue was applied to both sides of the plastic because steel retaining rings were to be glued on, to maintain the seal. The Front window was glued onto the steelwork first, then the ring, on the outside; both before the glue set. In the case of the cylinder, the ring was glued in first, then the cylinder to the steelwork, again, before the glue set. Despite the care taken in construction, the plastic parts inevitably have variations. Hence, before any gluing all parts were dry fitted, and the steel work given a final turning to tailor fit the parts.

A final pressure test of the whole assembly was made with liquid nitrogen (N_2) at TRIUMF. Again, the flask passed with no problems, and was ready for use in the AHEAD polarimeter. It has survived at least four experimental runs (cooldown, filling, continuous running, and warm-up) for the Saclay calibration experiment, so this project can be considered successful.

Chapter III. The Monitor System

The monitor system was designed to remotely monitor the functioning of the AHEAD LH_2 target system. The target system, schematically represented in figure III-1, consisted of the target flask, cryogenic system, vacuum insulation and vacuum pump system, all of which had to be monitored. Liquid level and pressure were main concerns in the flask; the temperature and compressor status for the cryogenics; vacuum level at various locations in the insulation and in the vacuum system; the status of various valves in the vacuum system. In all, 14 gauges and 48 switches or contacts had to be monitored (see Table III-1). A computer controlled monitor system reduced the necessity to physically inspect or read the devices being monitored, and could automatically and reliably log information.

1. Requirements

One of the most important requirements for this Monitor system was that it should not require a program expert. That is, programs should be easy to implement and modify by someone not very familiar with the system. A system expert will not always be available, and it may be necessary to make changes to the programs without him. This immediately eliminates any system requiring machine language, or similar program language.

Since it is meant to be a monitor, the system must be able to acquire, process and output data in a reasonable length of time.¹ At the time of this writing, there were 14 analog and 48 digital data points to be monitored. As it turned out, processing this amount of data was a fairly lengthy task. The target was controlled by an automated mechanical control system, capable of handling alarm situations, but the monitor had to be able to react fast enough to record the alarms. This meant that scanning data for alarms

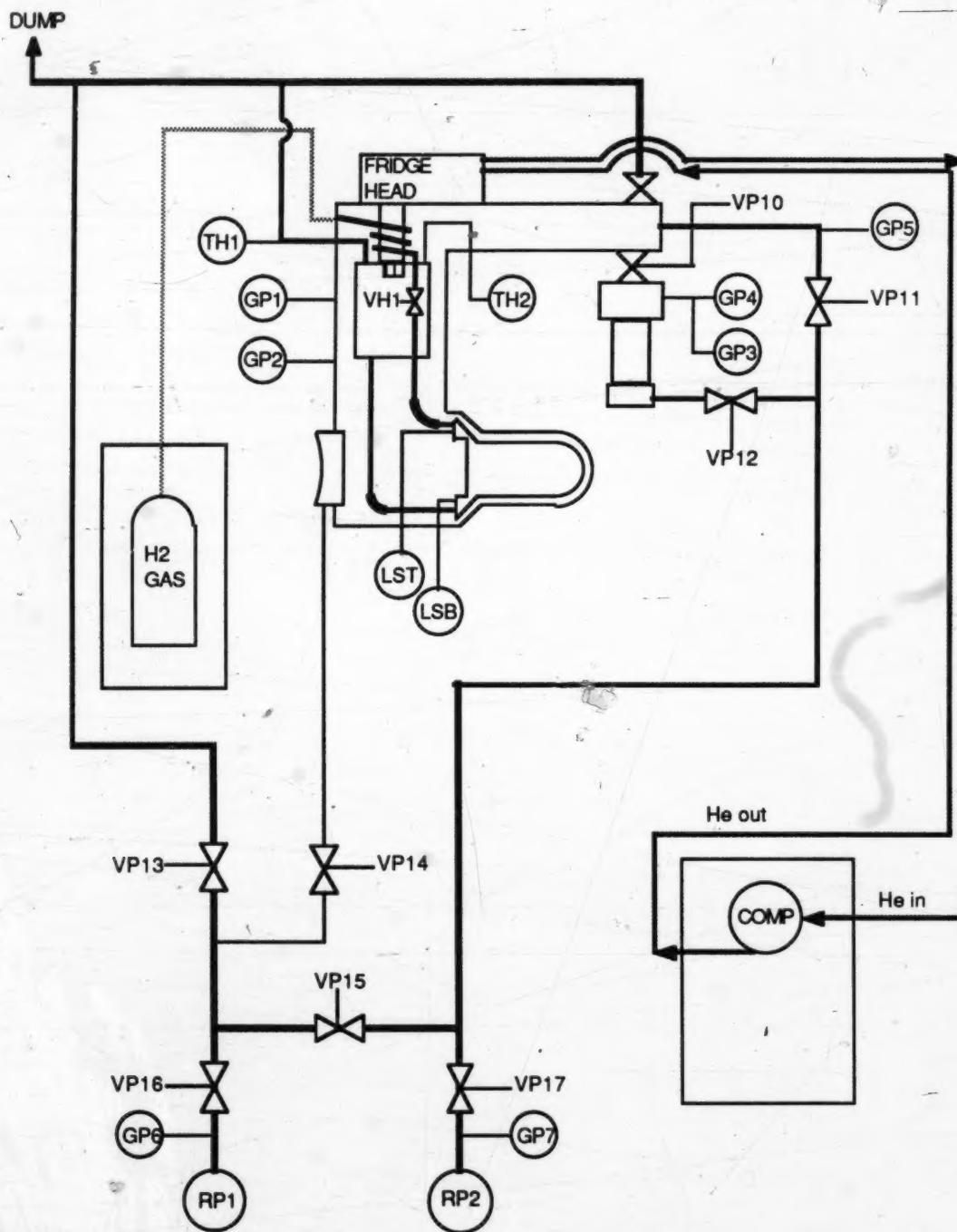


Figure III-1: Schematic representation of the AHEAD Target System and data points.

took highest priority. There were two ways of doing this. Use a fast personal computer (PC) (i.e. a 16-bit computer) and devise routines that handle processing with high emphasis on scanning for alarms. Alternatively, use a dual processor system; one machine to acquire and scan the data for alarms, and transmits the data for the other to process for output (see figure III-2). The latter was chosen because this separation of tasks was convenient and easy to work with, and most importantly, because scanning deadtime would be low.

2. The Two Computers

The front-end which satisfied most of the requirements was the Control Microsystems Stand-Alone-Front-End 8000 (SAFE 8000).² It uses a 65C02 processor which runs an extended version of BASIC. The physical design is based on NIM-type crates. The computer resides on two cards on the side of the 19" rack mount card cage. There are 14 slots available for peripheral cards, which are automatically identified when the computer is powered up. The acquisition cards used for the AHEAD monitor are: 1-Model 8531A 16 channel programmable 12-bit differential Analog multiplexer, 1-Model 8551 12-bit ADC (Analog to Digital Converter), 2-Model 8604A 32 channel digital input, 1-16 channel digital output.

Peripherals are controlled by concise commands in SAFE BASIC. These commands require very little in support programming, allow various options, and, as per high level language philosophy, are function-identifiable by name. The primary features used were: direct I/O, priority interrupts, multitasking, and timer delayed interrupts. Direct I/O can be carried out by commands as simple as AIN(c) (Analog INput channel #c), DIN(c) (Digital INput channel #c), and DOUT c,b (Digital OUTput on channel #c, bit #b), and were used to scan and acquire data. Priority interrupts were used to handle alarm situations in a hierarchical manner. Multitasking in this case was not true

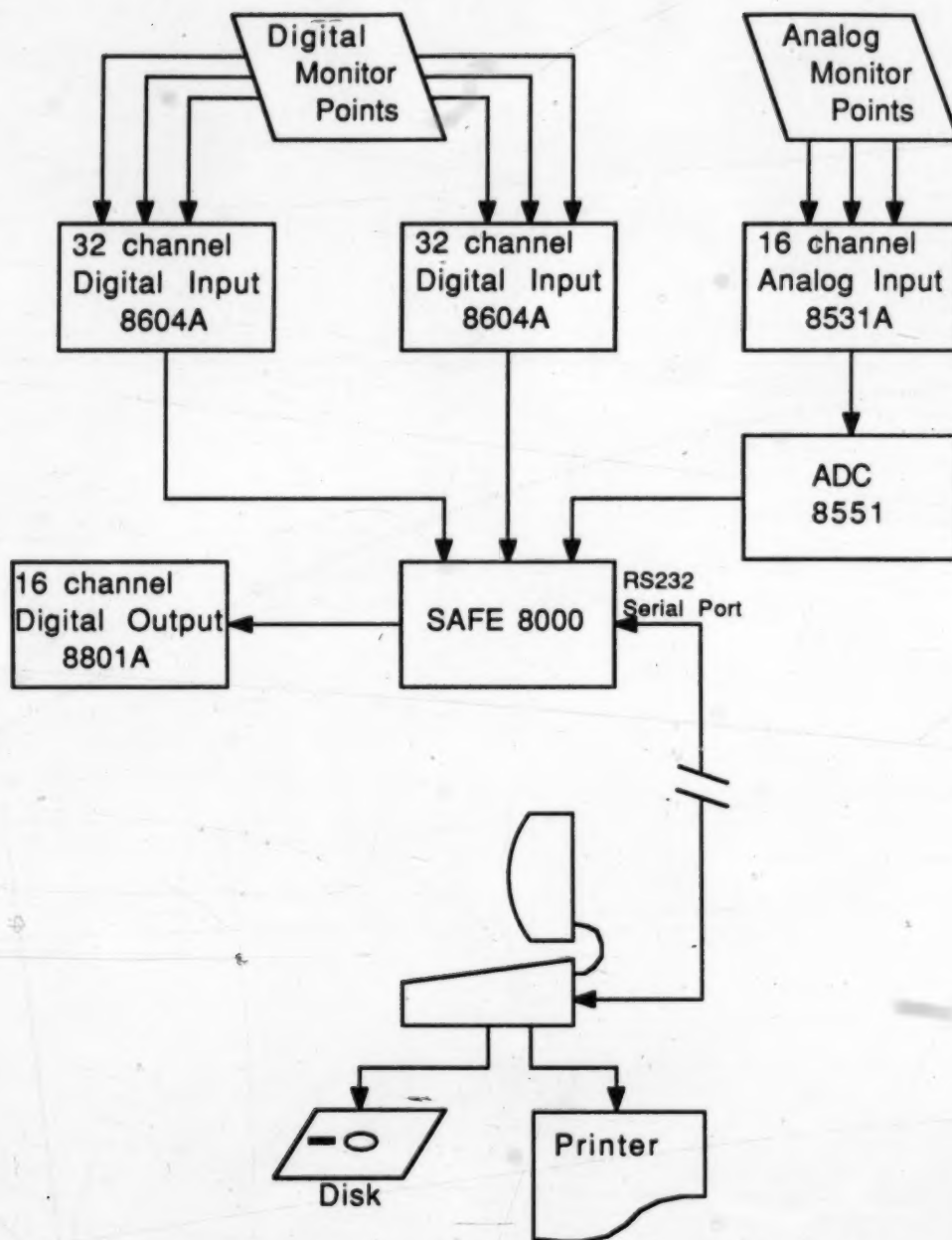


Figure III-2: Schematic of Monitor System Hardware

multitasking, but was similar to subroutines. It differs from subroutines in its flexibility. Specific routines do not have to be called; the next routine in the tasking queue is executed. There can be up to 32 tasks in a multitasking queue. As a result, calls for tasking can be placed anywhere without concern for which task gets executed. This also facilitates scanning of different cards in an alternating fashion, as opposed to sequentially. This allows a certain degree of a desirable simultaneity in processing.³ This feature was particularly useful when operator input was demanded, during which several hundred data points can be scanned. Timer delayed interrupts were used to determine "no response" alarms, that is, alarms that have not been resolved within some duration. These timers (numbering 128) worked in the background, hence hardly interfered with main routines. It is emphasized that each of the above functions was typically implemented by one command, i.e. peripheral management was built into the command.

Contact with the outside world was via one of two programmable RS232 ports. Either one can be chosen as the Master (or command) port and can be set to various configurations and speeds, by hardware or software. The monitor was run and edited with the master port configured at 1200 baud. Furthermore, the ports are buffered and can be timed out. The former ensures no loss of incoming data, the latter prevents prolonged communications hang-ups.

Programming and data processing was done on an Apple IIe equipped with Super Serial card (RS232), Extended memory (total 128K)/80 column card, monitor, 1 disk drive, and 1 printer. Programs on the Apple were written in BASIC (Applesoft). This part of the system had almost all of the problems in the system, primarily with the RS232, the memory, and even its software architecture. The Apple Super Serial Card was not buffered, hence loss of data was a serious threat. To compound that problem, Xon/Xoff (enable/disable transmission) protocol was user software dependent. The two problems resulted in frequent loss of the first character(s) after a carriage return (i.e. of every line). This was in the Terminal mode of the Serial interface, which was used to program the

SAFE 8000. In the PC program, one is faced with having to make multitudes of peripheral switchings (using IN#n and PR#n BASIC DOS commands) because the Apple can input/output to only one device at a time. Hence, the program is littered with these peripheral access commands, slowing down the PC's response time. These problems had to be accounted for, corrected or circumvented mostly by the Apple program, though the SAFE program frequently had to use timing loops to account for delays in response from Apple. To summarize, the Apple peripheral system was primitive.

The architecture of the Apple II memory created other cumbersome problems (see figure III-3). When using BASIC, one has very little control over how memory is used, and it was discovered that the PC program violated the High Resolution (Hi Res) graphic memory. This was because Hi Res was located 8K into free RAM (Random Access Memory) and extended for 8K (free RAM starts on Page 8 in memory and Hi Res started on Page 32, where 1 Page = 0.25K). Since the final PC program used 17+K, and Hi Res graphics were used in the monitor, half of the program was inevitably destroyed, causing a crash. The program was moved above Hi Res memory. This area contained 32K. Processed numerical data required 10K. Combined with the program, this left very little room for data storage, and the alphameric input buffer. The latter was a source of substantial irritation. Whenever any alphameric variable is inputted, it is allocated cumulatively at the top of memory. Since data was transmitted as alphamerics, as a measure against transmission error hang-ups, this quickly filled a substantial part of memory. This had to be cleared after every transmission (FRE command), resulting in a three to five second hang time, a fair wait.⁴

The data storage problem was solved by the extended (Auxiliary) memory. Since this memory was useful only for machine language programs and data storage,⁵ it was used heavily for the latter. This was done by POKEing data into the 8K dead space below Hi Res memory, where it could then be transferred to the Auxiliary memory. In 8-bit machines like the Apple, POKEd data must be 8-bit (0-255), but Analog data ranged

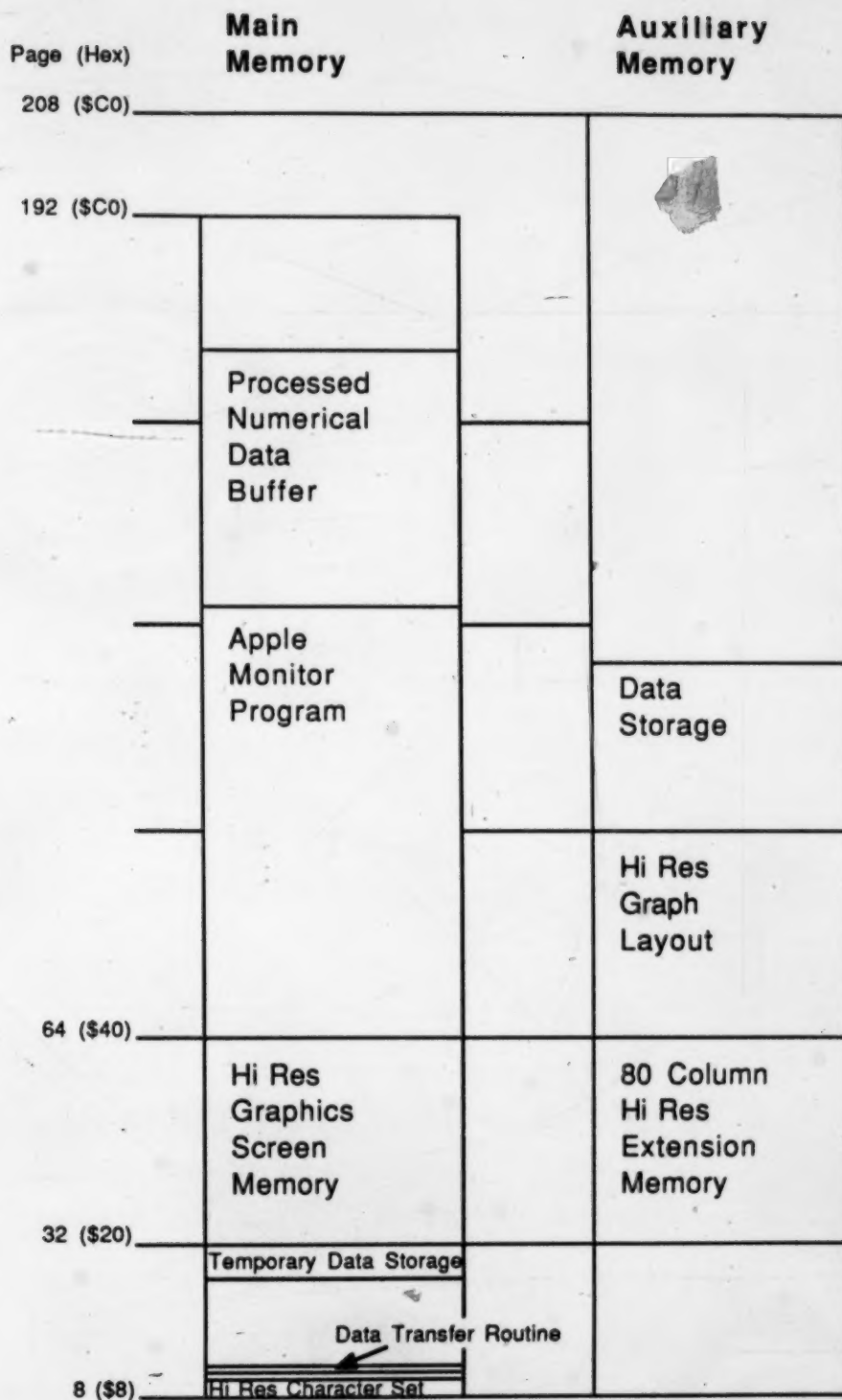


Figure III-3: Memory Map of Apple IIe

from 0-4095, so had to be converted to two 8-bit words. Digital data was 8-bit, so no conversion was necessary. The 8K space was also used to store the Hi Res character set and the only machine language subroutine written for the system - the data transfer routine (see program listing E.5, Appendix E). The latter incorporated a routine included in the Extended memory card and was the only means to transfer data from main memory to auxiliary memory, and vice versa.⁶ It also proved to be useful for Hi Res graphics. The graphs in Hi Res had to be refreshed occasionally, and by storing an empty graph in Auxiliary memory and calling the data transfer routine, the refreshing could be done in a small fraction of the time it takes to totally redraw the graphs.

The monitor system functioned parasitically with the target system control panel. The panel had overall control of the target, including hardwired alarm interlocks. From this panel, the monitor system was allotted panel outputs which it could read, and some control inputs with which the monitor could influence the control system by vetoing panel outputs.

3. Program Details

Experience has shown that modular program design is best. This allows a great deal of flexibility in logic and in modification of programs. Each "module" represents a general function of the monitor (see figure III-4), which contains more detailed sets of instructions to make that component work. If a problem occurs, it is usually isolated in one "module". And if a change must be made, the others are not seriously affected (if at all). Placement of the "module" in the overall monitor algorithm may also be easily changed. So this principle was used as a basic philosophy for the monitor. Modularity was easily effected in the SAFE using its multitasking and interrupt system. The Apple had more complex tasks to perform and problems to overcome (requiring a great deal of effort and program space), but they were also separable and modularized.

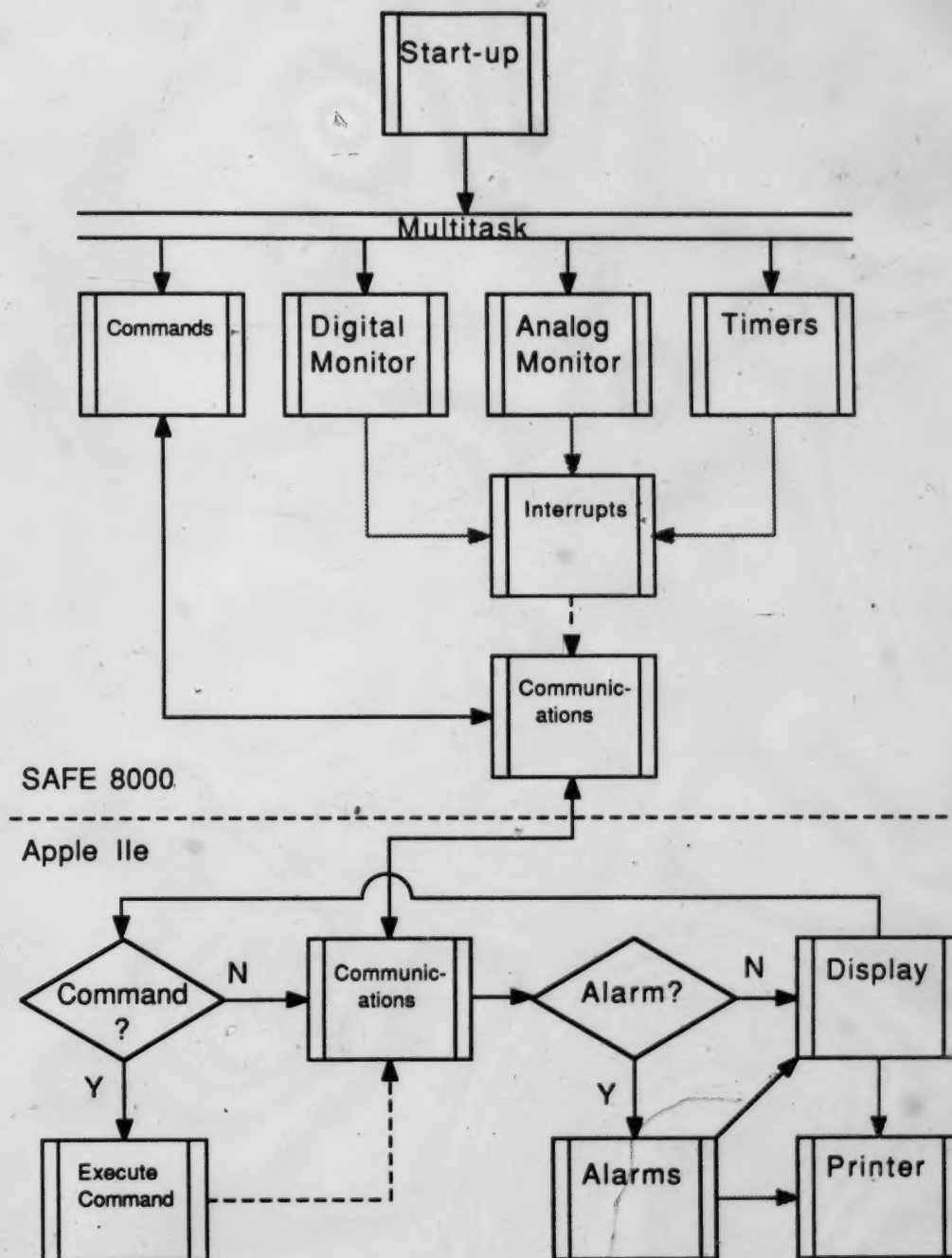


Figure III-4: Schematic of Monitor Program Logic

BASIC was used because of a lack of high level languages that could fulfill two criteria simultaneously: 1) high level so that reading the program listing should require a minimum of decoding; 2) quickly and easily modifiable in the event that a modification is called for during a run. Just about any high level language satisfies 1). But compiled languages would have proven disastrously cumbersome: modification would require loading an assembler, making the changes, compilation, then loading the modified code. BASIC is the most common language that satisfies 2). It is an "interpreted" language. The machine reads and executes the code, word by word, as it was programmed by the user. As a result, it is inefficient and slow. But its virtue is that modification only requires the change to be made, then the program started immediately. Actually, speed was of little concern for SAFE BASIC. This specialized version of BASIC was much more powerful and had much fewer (if any) quirks than Apple BASIC, hence ran fast enough for monitoring purposes. It took about one second to cycle through all 62 data points - inputting, decoding digital data, checking alarms, and checking timers.

The Apple presented a different situation. It was bogged down by the volume of processing and by the corrections and circumventions resulting from hardware and architecture deficiencies. There were many complaints (justifiable) that the delay between updates was too long, typically twelve seconds for a full scan, but little could be done to alleviate the problem short of writing in machine language or using a faster computer (e.g. 8800 or 68000 series processors).

3.a. SAFE

The SAFE program (see program listing E.1, Appendix E) was separated into three levels of execution: Communications, Alarms, and Scanning. Scanning simply involved inputting each channel and comparing it to preset alarm set points (see figure III-5a), and checking timers for overdue response from the control system on a previous

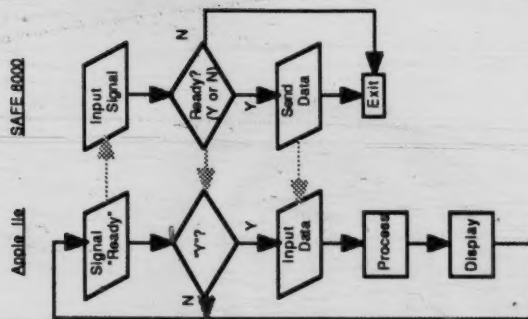


Figure III-5d: Logic of Communication Routines for Data Transfer

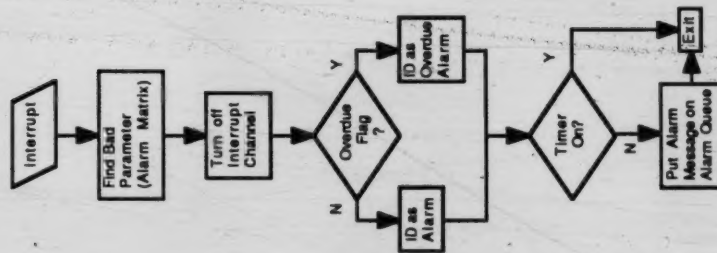


Figure III-5c: Logic of SAFE 8000 Interrupt Routine

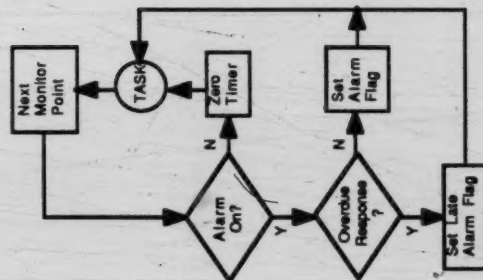


Figure III-5b: Logic of SAFE 8000 Timers Scan

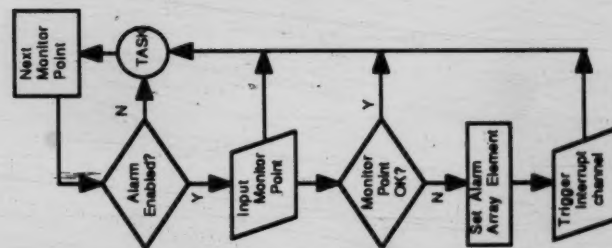


Figure III-5a: Logic of SAFE 8000 Monitor Point Scan

alarm (also an alarm condition)(see figure III-5b). These are done in three multitasked loops. One loop checks analog data (lines 610-700), the next checks digital (lines 710-1210), and the 3rd, timers (lines 480-600); each loop executing one cycle then switching to the next loop; the three loops also being looped, unless an interrupt is triggered. This simulates three simultaneous independent loops. This independence allows each loop to be executed at different paces and to have different lengths, yet to be executed almost simultaneously.

The algorithm for analog and digital scan is basically the same. The priority level is checked first. If its value is greater than 23, then the alarm for that data point has been disabled and a new TASK is called. Otherwise, the data point is compared to its set points. If it passes, a new TASK is called. If it fails, an alarm flag is checked to see if the present alarm is consecutive to a past alarm. If not, the flag is set. If so, this implies a prolonged alarm condition which the timer routine handles. Both cases are followed by placing the alarm on a priority queue. The priority queue actually consists of a 24 x 5 array. This allots 5 interrupts for each of the 24 priorities. Each bin in the array contains the variable number of the problem data point, i.e. its identity. Then the priority interrupt is triggered. This is done by setting a digital channel in slot 0 or 1 on. These slots, in addition to being able to do normal I/O, are directly connected to the interrupt system of SAFE. And since SAFE is capable of setting digital input internally, the priority interrupts are triggered by setting the corresponding digital channel or bit on. Finally, a new TASK is called.

The digital scanning routine differed slightly from the others in that some data points were treated more specifically. Valves had two sets of contacts: one when opened, and one when closed. This was to ensure that a valve was not left half open. Hence, scanning these data points required that both contacts be cross checked. Since these valves had either 2 or 3 data points associated with them, it was not possible to use a straight loop. Instead, a subroutine was called, with the digital variable number as a

parameter, for each valve (lines 840-930). This subroutine compares the 'open' with the 'closed' contact for a half open valve. Then the 'open' contact is compared with a set point. The half open valve could be more serious than an improperly set one, so it was given higher priority.

Secondly, the alarms from the pressure gauges were not scanned, since the analog data from the gauges was being monitored. In fact, it was simpler to check for these alarms at the PC.

Digital data needed a further refinement in general. The digital card reads eight data points at once, considering them as one 8-bit input, so that data from that card is in 8-bit words, each word coming from a channel. The data point then had to be extracted from the appropriate word. There are a few ways to do this. The one chosen was to breakdown the variable number into the corresponding channel number and bit number on the digital card. The channel was read, then that word was EORed (Exclusive ORed, a bit-by-bit EOR included in SAFE BASIC) with the bit pattern for the data point on that channel. The normalized result is then compared with a set point (e.g. lines 740-760).

Another departure from the standard alarm scanning routine concerns the printer dump. Since SAFE has timers which operate independently of programs, one was used as a timer delayed interrupt to regulate printer dumps. A countdown timer is set to output to an input channel in slot 0 (the interrupt slot) which has an interrupt routine specifically for coding a print dump trigger message for Apple. Priority level 1 (channel 0, bit 1) is reserved for this interrupt.

One interrupt which is global in effect is the emergency shutdown. It has priority level 23 (highest) and is triggered, not by setting it on, but resetting it (i.e. set to 0). It does not have any data points currently associated with it, but it was envisioned that it would be attached to the data points associated with the cryogenic system and electrical power supplies.

An alarm condition that does not follow any of the interrupt scheme is the

Autostart alarm. This is more an alarm than the print dump but less so than any of the other alarms. When SAFE is powered up, it outputs the usual system status, but it also automatically commences program execution. This is convenient in mid-execution if SAFE is inadvertently, or otherwise, shut down, then restarted. Normally, when the system is started up, SAFE is commanded to start execution at line 50. If, in the case of an autostart, execution starts at the first line (line 10), the command flag MC\$ is set to "E", as opposed to "M" for a normal start, and bypasses some initial program lines. SAFE program execution goes as normal until the command flag is checked, which then directs execution to the Autostart alarm routine. Due to the low key nature of this alarm and the brevity of its treatment, this routine remains at the scanning level and puts the following coded message on the output queue: 1) "I"(interrupt) + "E"(data type = Autostart alarm); 2) "SAFE AUTOSTART"; 3) time; 4) date; 5) "\$". This message is put on the output queue, then it returns to main execution. This Autostart feature also allows Apple to continue with a minimum of disruption, which will be discussed in Sec 3.b.

The timers algorithm is the same for analog and digital data. The alarm flag is checked, and if not set, timers and second alarm flags turned off (or kept off if there was no previous alarm). If it was set, and the timer run down, a second alarm flag is set. In any case, at the end a new TASK is called. As noted previously, a complete uninterrupted sweep of data usually takes about 1second.

If an alarm situation is detected, it is noted and an interrupt is triggered. There are three reasons for triggering an interrupt and not treating the problem directly.

1) The interrupt system easily organizes alarm treatment, facilitates priority changes, and is open to specialized or customized alarm treatment without major global modifications to code. There are 24 levels of interrupts. Several alarms with common characteristics and alarm handling can be grouped together in one priority interrupt level. Vacuum pressure gauges were attached to levels 19 and 20; target temperature and pressure were given higher priority, 21 (see Table III-3). These levels are assigned to

each data point at the start of the program (lines 4340-4400). Since each of these levels are referred to by number, changing priority for a data point is simply done by changing a number in the code, or by command during execution.

2) It is possible to directly trigger alarms using a digital input card in slot 0 and 1, so keeping this option open is desirable.

3) There is a short delay between trigger and response, about 0.1 seconds, but long enough for 2 to 5 more data points to be scanned, in case of a more critical problem (and hence, more probably the cause).

When an alarm is triggered, it is handled by routines specified by its priority level. While each data point may need special treatment, the algorithm is always the same: ID the offending data point, perhaps treat the problem, code information for output, and place that information on an output queue (see figure III-5c). In reality, ID starts at the scan, where the problem data point is put on a priority queue and the interrupt triggered. At the interrupt level, these data points are treated one by one according to the priority interrupt and the priority queue. SAFE always processes the current highest priority first, sometimes interrupting one interrupt in favor of a higher priority one. The program then calls a subroutine specifically defined for a certain interrupt at the program start up (lines 160-390). Then each bin of the priority queue for that level is checked for alarms and all alarms at one priority level are taken care of before exiting the subroutine. If an alarm is found, the offending data point can be identified by the contents of the bin, placed there by the scan routine.

Once the alarm is identified, its digital interrupt bit in slot 0 or 1 is turned off. The second alarm flag is checked to select an interrupt code for transmission. "I" was used for an ordinary interrupt, and "L" for second alarm. Next, the timer is checked. If it has not run down, the interrupt transmission is cancelled because the proper interval for what is considered a prolonged alarm has not yet expired. This interval was set at the start of the program, and is modifiable, by command, during the run. If the timer has run down

or it had not previously been started, in the case of a first alarm, then the timer is restarted, followed by the coding of information.

The coding is formatted as follows: 1) interrupt level ("I" / "L" = first/second alarm) + data type ("A"/"D" = analog/digital), 2) data point name, 3) data point number, 4) data value or message, 5) time, 6) data, 7) "\$" terminator. The data point name may be altered. For example, vacuum pressure gauges had set point alarms built-in that could be changed without the corresponding change in SAFE. So an interrupt routine (lines 2280-2530), specifically for the vacuum gauges, checked the corresponding digital (alarm) data point for an alarm. If both, analog data and alarm, indicated an alarm condition, a normal alarm was transmitted. But if only analog data indicated an alarm condition, the variable name was prefixed with "SUSP", meaning a suspected alarm condition.

A message replaces data value in two possible cases. Each valve in the vacuum system had at least two data points: set point alarm and half open alarm. A simple code could not be devised for the half open alarm so a message, "Half Open" was used (lines 2170-2270). The other situation concerns data points directly involved with the H₂ target cell. These are target pressure (TH1), target temperature (TH2), vacuum (GP1), and gas monitor. Diagnostic logic (see figure III-6) had already been devised around these data points, and since it yielded more meaningful information, the logic was incorporated into a separate interrupt routine for the target cell (lines 1930-2163).

The information for each alarm must be put on an output queue simply because the Apple, unequipped with peripheral interrupts, will most certainly be unready for transmission. Hence, the final step must be handled by the Communications routine.

Communications (lines 1220-1540) also works on a priority interrupt basis, having the highest priority levels, 24 and 25. This routine handles two functions: transmitting information to, and accepting commands from the outside world. Both require a signal from the Apple to be sent, indicating that it is ready (see figure III-4d).⁷

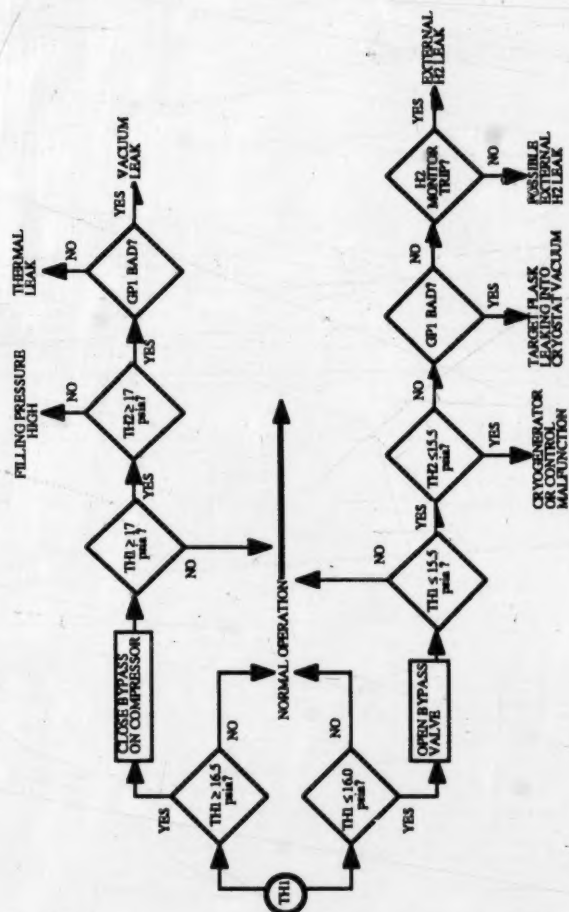


Figure III-6: Diagnostic logic for target cell alarms.

The signal is a one letter code (though more than one is technically feasible), either a transmission request or a command code. This triggers a communications interrupt, causing the program to go to the Communications routine. Here the incoming code, held in the buffer, is read and decoded. The actual codes are detailed in Sec 3.b.

SAFE responds with a request "received" code ("Y") or a negative ("N"), depending on whether or not the last signal was recognized and SAFE is ready to transmit. The only case in which SAFE would not be ready to transmit is for normal transmission if a timer is not run down, in order to maintain some semblance of periodicity. If it is ready, then the alarm output queue is checked. Alarms are transmitted one for every transmission request. If the queue is empty, then all digital data is sent, and at the next transmission request, all the analog data (lines 1380-1540).

Data transmission is formatted⁸ as follows: 1) interrupt code + data type; 2) data word;... n-1) last data word; n) "\$" terminator. The interrupt code indicates whether the following data is data, alarm, or special (see also Sec 3.a. and 3.b.iv). Data type indicates whether the data is from analog or digital cards. Each of these two pieces of information are represented by a single letter code, and together form one "word". This is the general format of any transmission from SAFE to Apple (e.g. compare with Interrupt coding). Each data word is transmitted on one input line to avoid fixed-formatted inputs in the Apple program. Also, due to the frequent loss of the first character for every line input, data words were padded with spaces. In the case of transmissions of data on the output queue, everything was stored in one character variable and separated by ASCII carriage return codes (CHR\$(13)). When outputted, the Apple sees this as the usual series of separate line inputs of one word each.

Commands from the Apple can be executed on two levels. Immediate execution is used only if there is no user interaction involved. This is carried out at the Communications interrupt level, i.e. there are subroutines in the communications interrupt subroutine to handle these commands. If, on the other hand, the user must input data

(i.e. from the keyboard) directly to SAFE, execution occurs at the scanning level. That is, because multitasking in this machine does not function in interrupt called routines, the Communication interrupt routine must be exited. And whenever a user input is called for, the scanning continues during the waiting period, between keystrokes, through multitasking. Data point parameter modification, clock reset, and manual digital control are handled this way.

3.b. Apple II

The program for the PC end of the monitor system is divided according to the screen output format, with alarm and other interrupt routines at higher levels (see program listing E.2, Appendix E). Each screen has the same program structure: Check keyboard buffer for user command, check for "transmission request received" signal from SAFE input data, and output to screen.

User commands are single character codes which are defined for the screen in use. A menu at the bottom of the screen lists the codes and corresponding function. With the exception of the command mode, these functions are mostly for switching to another display mode, or to terminate the program(s). The modes are shown in figures III-7.

If a command requires that SAFE perform a function, or if there is no command, a command code, or transmission request signal ("S"), is sent to SAFE. A 'received' code ("Y") is expected and its absence voids the subsequent function. In the case of the command code, specific routines are run after the 'received' code. For ordinary data transmission, after the 'received' code, the input routine continues as usual.

The data input routine is used as a subroutine (lines 20-690), callable by any of the routines handling the screen modes. It uses the BASIC INPUT function to read data one word at a time, for an unlimited number of words. This function always displays inputted data on the screen so that despite closing up the screen window as much as

possible, there is always a flickering at the top left hand corner of the screen. That is the incoming data as the INPUT function reads them. This also means that line 1 cannot be used for display purposes (except for graphics, which overlay the text screen). The input routine is placed at the beginning of the program to ensure that the interpreter executes it promptly, and is combined with the command routine.

While the amount of data processed by the PC program was fixed, the amount actually inputted is not limited. This program is more "customized" than the SAFE program out of necessity. Some of the data points needed special treatment in order to be presentable as output. The free format of the input routine was necessary to avoid the liability of hang ups due to RS232 misbehavior (e.g. loss of data). Second, it allowed the same routine to be used for the different data types; digital transmissions contained eight data words compared the analog's sixteen. While transmission faults would result in erroneous data, it is believed that it would be obvious upon inspection because it would result in a discontinuity in data.

Everything is inputted as Alphabetic strings. This allows the same routine to be used for any kind of transmitted information (hence reducing response time to incoming transmissions), and again turns fatal transmission errors into nonfatal data errors. Unfortunately, this creates a huge volume of alpha data which must be cleared from the alpha buffer. This takes three to five seconds and is necessary to do at every transmission because the remainder of free memory is not large enough to accommodate any more input than that from one transmission.

Once inputted, the data is converted to numeric values. Analog data is placed in temporary memory and also converted from a 12-bit word (0-4095) to two 8-bit bytes and placed in the 8K 'dead' space for later transfer to Auxiliary memory (then to disk). It is also plotted on graphs hidden in Hi Res memory. This is to maintain continuity of the graphs, or alternatively, to keep screen change durations low.

Digital data is received as 8-bit words (each bit represents one digital data point) so

it is not necessary to break it down for the 8K space, but for output purposes, it must be broken down into constituent bits and put in temporary memory. This process would have been trivial to perform as a Machine Language subroutine, but it was thought that this would infringe on the 'high level' condition. Once in temporary memory, the data is ready for processing and output.

The 8K 'dead' space accommodates data for 40 full transmission cycles (i.e. 40 digital bytes and 80 analog bytes). At certain intervals (20 cycles), the lower portion of that memory is transferred to Auxiliary memory (via a machine language subroutine). Then the upper portion is shifted into the lower portion. This is to make room for new data, and, at the same time allow continuity from old data to new data on data vs. time plots (i.e. crude scrolling). The data in auxiliary memory accumulates until a limit is reached (six memory transfers, equivalent to about 25 minutes worth of data), and the contents transferred to disk, if enabled, or cleared.

3.b.i. Screens

Screen output options are: Alphameric, graphs, horizontal bars, and raw data.

The alphameric screen converts data into physical quantities (e.g. for pressure gauges, Volts to Torr), and is divided into four sections on the screen (see figure III-7a). The Top section contains general alarm status, e.g. power failure, gas failure. The section below it contains information on the vacuum system outside the target system, e.g. roughing pumps. The lower left section contains information on vacuum inside the target system (but outside the target cell). The lower right has information on the cryogenics system.

The 80 column mode of the extended memory card could not be used here or in any text output screen. It required that a peripheral slot be continuously open. Since only one peripheral slot could be opened at a time and at least one peripheral (the RS232) was


```

ALARM      N2 OK      115V ON
GAS OK     AIR OK     +6V ON

RP1  GP6 OK  VP16      VP13  VP14
A ON x.xxE-xMB A ☒ C VP15  A ☒ ☒ ☒ C
              O ☒ C

RP2  GP7 OK  VP17      VP12  VP11
A ON x.xxE-xMB A ☒ C  A ☒ C A ☒ C

DIFF      VP10  VH1      FRIG  COMP  COOL /
A ON      A ☒ C A ☒ C  A ON  A ON  BYPASS
              VR5 LOCAL, TH1, TH2

GP3 OK      GP4 OK      TH1 xx.xxV (TGT)
x.xxE-xMB  x.xxE-xT    TH2 xx.x K (VB)
GP2 OK      GP1 OK      TC1 (CH) xx.xxV
x.xxE-xMB  x.xxE-xT    TC2 (FL) xx.xxV

GP5 OK      LST NO xx.xxV
x.xxE-xMB  LSB ON xx.xxV

NUMERIC      10:55:20 25/08/86
X-EXIT PC C-CMD G-GRAPH H-H.BAR Q-QUIT

```

Figure III-7a: Representation of Alphameric Screen. Valves and other digital data point states are highlighted (Reverse field).

TH1		TH2		VR5		GP1	
GP4		LSB		LST		TC1	
TC2		N/U		GP2		GP3	
GP6		GP7		GP5		N/U	

```

10:55:22 25/08/86
X-EXIT PC C-COMMAND N-NUMERIC Q-QUIT

```

Figure III-7b: Representation of Graph Screen

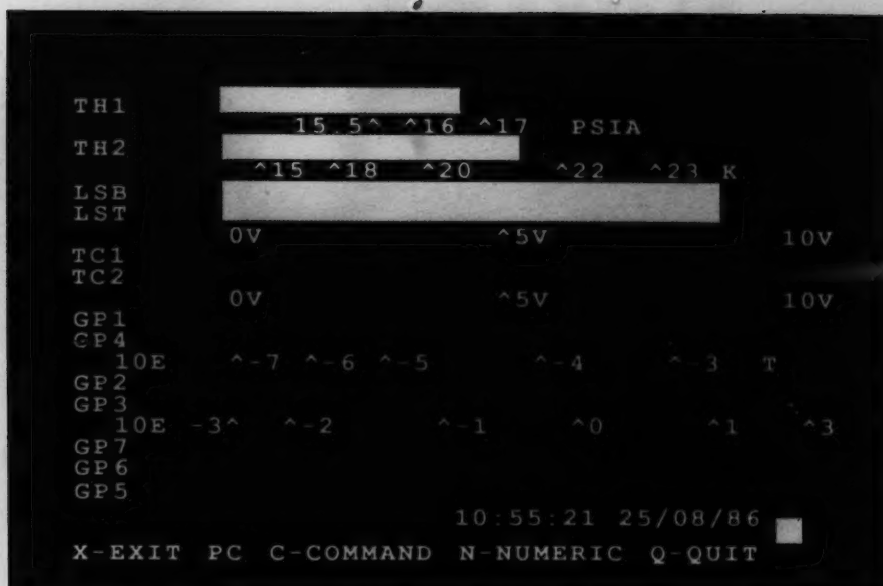


Figure III-7c: Representation of Horizontal Bar Graph screen

Figure III-7d: Representation of Raw Data Screen.
"D" Suffixes indicate disabled alarm.

X-EXIT PC, LEAVE SAVE RUNNING
G-GRAPHS
N-NUMERIC
H-HORIZ. BAR GRAPH
R-RAW DATA
M-MODIFY PARAMETER
S-SET TIME &/ DATE
Q-QUIT
P-PRINT DUMP
T-TOGGLE PRINTER ENABLE (FROM 1)
I-SET PRINTER DUMP INTERVAL
D-TOGGLE DATA DISK (FROM 1)
F-DISK FILE DIRECTORY
A-TOGGLE ALL AVAILABLE ALARMS (FROM 1)
O-DIGITAL OUTPUT CONTROL
B-TOGGLE 'BEEP' (FROM 1)

COMMAND

10:55:23 25/08/86

Figure III-7c: Representation of Command Screen

repeatedly opened and closed, the 80 column mode could not be practically kept open. The secondary reason was that 80 columns pushed the limit of resolution on the monitor being used (a somewhat venerable device).

The graph screen displays 16 analog channels in graph form on one screen (see figure III-7b). These are updated continuously, regardless of what screen is currently up (i.e. in the background). The plot is effectively channel vs. time, with fluctuations in time because the timing is not meticulous. Unfortunately, to save memory and processing time, it was necessary to put all 16 channels on one screen, so that each graph had 40 x 40 pixels available. Hence graduations were left unlabelled due to poor resolution (consider that one resolvable character should take up 5 x 6 pixels). Note that this also means that the analog data was approximately incremented by 100's, but at least there were 40 measurements displayed for one data point - equivalent to ~50 minutes. On the far right hand side are four clusters of horizontal line segments. These indicate the status of the digital channels in the two SAFE digital cards, but are pretty well useless unless one can identify them. These data points are identified in Table III-1b.

Horizontal bar displays analog data as horizontal bars, with 30 increments full scale (see figure III-7c). The graduations were labelled for this screen, and represented physical quantities (e.g. Torr for the roughing pumps).

The raw data screen displays data in the form the SAFE 8000 received them (see figure III-7d). That is analog data was in voltages, digital data as 0/1 (off/on). This screen is basically for diagnostic purposes. A particular situation in which it was of great help was in correlating channels with data points, and identifying any faulty lines or connections. It also aided in remotely calibrating the set points for the target temperature and pressure gauges (as opposed to going to the target control panel standing in a radiation restricted area).

3.b.ii. Processing

Each screen has its own set of variables, particularly for data. This reduces limitations on data processing (e.g. cutoffs) by leaving original data intact for additional processing (or carrying over to another screen). This translates to more flexibility because data processing can be done in stages. Furthermore, this allows grouping of common calculations instead of explicitly programming individual data point calculations, thus reducing program memory consumption.

Most of the processing is either simple scaling or digital decoding. Applesoft BASIC ORs are not bit-by-bit ORs (as they are in SAFE BASIC), so it was of no use in digital decoding. The digital decoding would also have been easy to implement in machine language, but to try to maintain the "non-expert" philosophy of the system, it was done in BASIC by comparing against and subtracting off powers of 2 (lines 410-450). Once this was done, screen output was merely a matter of using IF statements to set or reset an indicator. In addition, if the data point represented an alarm, it was a simple matter to incorporate some kind of alarm (presently, a flashing indicator).

The graphic type outputs (e.g. graph and horizontal bar modes) involved scaling to the appropriate increments, i.e.

$$O/P = (\#full \text{ scale increments}) \times \frac{\text{data}}{4095} \quad (III-1)$$

Analog information (0-10V f.s.) was converted to 12-bit data (0-4095) by the SAFE's ADC, hence the denominator is 4095. The full scale increments for Graph mode was 40, and for the Horizontal Bar mode, 30. The nonlinear correspondence of the devices to voltage was represented in the graduations.

There were two analog devices that required modifications of the above conversion. The H₂ pressure transducers, TH1 and TH2, control the refrigeration system, and in doing so they have short periods (about one minute) and limited voltage

variations. They never fall near zero either. Hence, to focus on a more relevant range, cuts were placed. Old code did not have to be overhauled, only the insertion of the cut definitions was necessary. This illustrates the usefulness of having independent screen routines (with their own variables) (i.e. modular design).

Alphanumeric analog outputs were complicated by the necessity of rounding off numbers, usually to three significant digits. BASIC does not support formatted output statements, and neither does it have digit suppression. Hence, round off had to be done explicitly to the variable in question. The algorithm used was:

$$V = [\text{INT} \left(\frac{\text{data} \times 10^{-s} + 0.5}{4095} \right)] \times 10^s \quad (\text{III-2})$$

where $s = \# \text{ decimal places} + 1$

$\text{INT}(\)$: BASIC floating point to integer conversion function

Where rounding off was not necessary:

$$V = \frac{\text{data}}{4095} \quad (\text{III-3})$$

sufficed.

Pressure readings were always in scientific notation because at first order, they are logarithmic. Here, INT does not perform the desired function as it did for voltage conversions. What was needed was to isolate the mantissa and exponent as two separate quantities. To this end, the common LOG of the fitted Pressure function was calculated: the integer part of the result was the power of 10, the decimal part was LOG(mantissa) (e.g. lines 1620-1630). The remainder of computation was the same as that for the voltage.

The pressure calculations on their own were not trivial, considering the Apple's capacity. The fitted equations were originally of the form:

$$P = 10^{m_1} \times 10^{[m_2 V + m_3 \exp(m_4 V)]} \quad (\text{III-4a})$$

or

$$\log(P) = m_1 + m_2 V + m_3 \exp(m_4 V) \quad (\text{III-4b})$$

where for GP1 and GP4, and for GP2,3,5,6,&7

$m_1 = -6.941$	$m_1 = -2.827$
$m_2 = 0.536$	$m_2 = 0.496$
$m_3 = 0.393$	$m_3 = 0.250$
$m_4 = 0.510$	$m_4 = 0.669$

The extra exponential corresponded to a nonlinear drop at the lowest pressures that the gauges measured. It was unfortunate that the relevant range of pressures covered both the linear and nonlinear regions. Exponentials take five to ten times longer to evaluate than the four primary operators. This further lengthened the delay between input and output.

Three linearizations were used in an effort to find a fit to calibration data for these gauges. At first a linear combination of exponentials was tried using the University of Alberta MTS (Michigan Terminal System) version of BMDP⁹ statistics software. A better fit was found using the above equation. Unfortunately, it was carried out on the Apple, at Saclay, because of a lack of familiarity with its resident mainframe. A linear least squares fitting program was used with estimated exponential parameters, i.e. it was not highly optimized. The calibration run in August 1986 use this second fit. After returning to Alberta, a third fit (see figure III-8) was performed resulting in a better fit and the program has been modified with it:

$$\log(P) = m_1 + m_2 V + m_3 \sin(m_4 V) \quad (\text{III-5})$$

where for GP1 and GP4, and for GP2,3,5,6&7

$m_1 = -6.941$	$m_1 = -2.827$
$m_2 = 0.536$	$m_2 = 0.496$
$m_3 = 0.393$	$m_3 = 0.250$
$m_4 = 0.510$	$m_4 = 0.669$

One other nonlinear device needed alphameric conversion. TH2 was used as a thermometer. Being primarily a pressure gauge, a conversion from V to pressure was

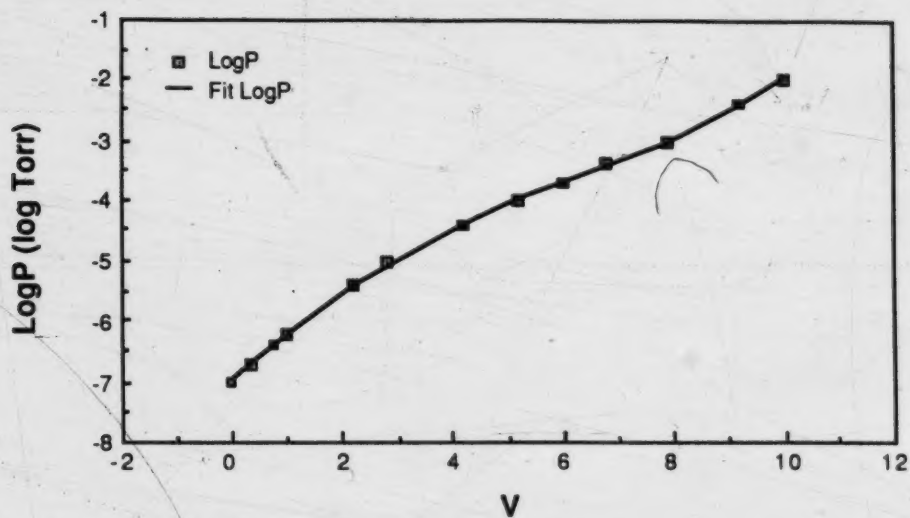


Figure III-8a: Log(pressure) vs. Gauge Voltage for vacuum gauges GP1 and GP4

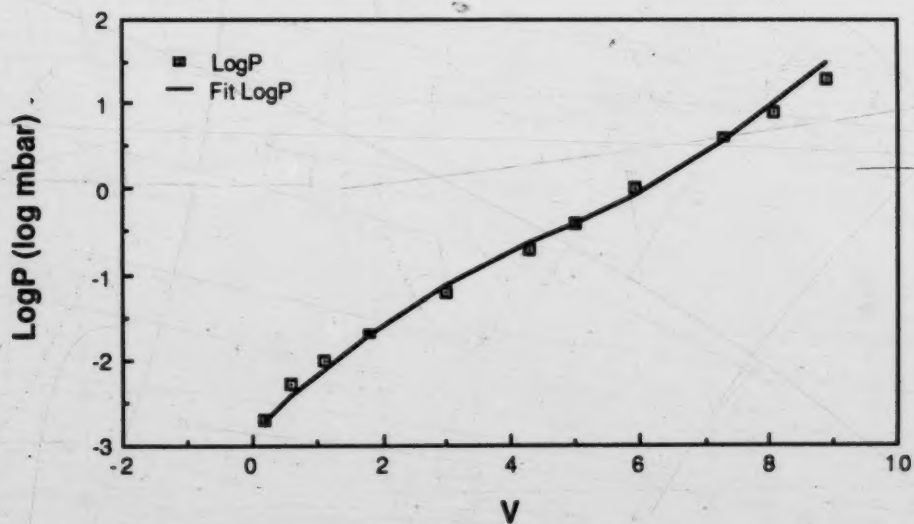


Figure III-8b: Log(pressure) vs. Gauge Voltage for vacuum gauges GP2, GP3, GP5, GP6, GP7

previously found, and was linear. The temperature dependence was not. A least squares fit indicated that a function of the form:

$$T = m_1(m_2(V - m_3))^{m_4} \quad (\text{III-6})$$

where $m_1 = 130.7$
 $m_2 = 7.8$
 $m_3 = 0.073$
 $m_4 = 0.1657$

to be highly satisfactory (lines 1450-1480).

3.b.iii. Commands

Each screen has a short menu of commands that switch to other screens. This is at the bottom of the screen because the Apple normally allots this space for text information. As mentioned before, a single key code from the menu executes the indicated function. The response is not immediate due to the priority placed on data processing. Command requests are handled only when the input subroutine is called (lines 60-110), partly to keep the program legible, but mostly because of memory considerations. Obviously, the more decoding routines, the more space used. Unfortunately, since Apple BASIC does not have active input functions, another means to check the keyboard for command input was needed. It was found that this information was stored at a specific memory location as a flag.¹⁰ This location was PEEKed and tested for input. If the flag was set then GET was used to input the keyboard input. Otherwise, the program continued normally.

There is another complication. A keyboard disable in memory could not be found, so that during data transfer, data could be corrupted by keyboard input. A keyboard disable/enable software switch would have been able to block keyboard input. This problem could not be resolved. The only alternative was to use a visual warning in the form of an on-line/off-line warning indicator, placed one line above the menu at the the far right hand side (lines 140 and 220).

There are two commands common to all screens. "Q" (=Quit) shuts down both

monitor computers. This should be used somewhat like a manual overriding shutdown. "X" (=eXit) shuts down the Apple only. A command code ("X") is sent to SAFE, which then goes to the "all available alarms" toggle routine, described later in this section, to turn off all alarms. When this routine is finished, it sends a "0" to Apple to indicate that the alarms are all off. SAFE can then continue to monitor processes while the Apple is taken off-line or out of the system. Both commands are followed by a request for confirmation to avoid inadvertent shutdowns.

The Command screen's main purpose is to allow the execution of special operations (see figure III-7e)(lines 3140-3940). "M" (=Modify parameters) sends a command code ("M") to SAFE that causes it to go to a routine that allows parameters to be modified. Data point names, priorities, and set points can be changed without stopping acquisition. As mentioned before, since user input is necessary, this routine must be run at the scanning level, so that multitasking can be implemented (scanning is not disrupted). The reason that SAFE is handling this function, not the Apple, is to avoid a potentially complicated redundancy and because of limited program space. Putting this routine in the Apple requires that all data point parameter information that is changed in SAFE must also be changed in the Apple. Most of this information would not otherwise be needed by the Apple. And whether the information is built in with the program, or if it is transferred from SAFE or any data storage, this information must ultimately be stored in active memory, of which there is little to spare.

"S" (~~=Set~~ time and date) is a function very similar to "M". It again sends a code ("S") to SAFE to go to a routine to change the current time and date at scanning level. This information is necessary for logging data and interrupts. Details of "S" and "M" are discussed in Section 3.c.

The monitor has facilities to periodically print out current data on paper and to store data on disk. The printer can be enabled and disabled at will using "T". This sets or resets a flag which is checked at the beginning of the printer routine. If reset to 0, the

routine is skipped over. The interval can also be changed using "I" (=Interval change). The timer is incremented in 0.25 minutes, and periods of days are possible, in principle. This latter function does not require information from SAFE hence the input part of the algorithm is done at the Apple. The time inputted is in minutes but the interval reset function in SAFE must use the number of 0.25 minute intervals. So Apple must multiply the user time by four (4). Once the new period is inputted and confirmed, a command code ("I") is sent to SAFE which goes to a routine at the Communication interrupt level, whereupon the new period is incorporated in SAFE. SAFE then echoes the inputted period to confirm the change which is displayed by Apple after the appropriate conversion.

Immediate printer dumps are also possible by "P", but the time will more than likely be incorrect. This is because the time the Apple has is updated only on reception of an interrupt (one of which is the periodic printer dump), which carry time and date. This function is executed simply by executing the printer dump routine in the Apple program (but bypassing the flag check).

The periodic disk dump enable can also be toggled by "D" (=Disk). Data is dumped into random access files on disk. It is possible to dump onto an existing open file (as opposed to a locked one) or a new file instead. If there are no existing open files, a new filename is prompted for. If a file is open, the user is asked if further dumps are to go to that file. If not, the file is LOCKed, as per AppleDOS, and a new filename is requested. In the case of new filenames, the program checks for duplicate names which would result in a disk error. In actuality, the disk directory is not checked, but a directory file created by the program is. This random access file keeps record of filename, lock status, and the number of records in the data files. The latter is important in preventing disk overflows, a disk error. There were two data files kept simultaneously. One is the main data file and the other is the interrupts, or alarm, file. The interrupts file uses the same name as the main data file except it is appended with ".I". This differentiates the

two data types by their different data formats yet makes their association obvious. If there is some question as to existing files, "F" displays the true disk directory.

It is also possible to enable and disable all available alarms at one keystroke using "A". This sends a command code ("A") to SAFE which uses a loop to toggle the enable bit in the interrupt priority codes for each data point. After all alarms are toggled, SAFE sends a "1" or "0" depending on whether the alarms were turned on or off. This keeps the Apple informed of the system alarm status.

There is a facility to manipulate selected switches using "O". A command code ("O") is sent to SAFE which executes another routine at the scanning level. The name and current value of each switch is printed out, and the user has the option of entering a new switch value (0/1) beside each one. A carriage return is entered if no change is desired for a switch. This protocol was inspired by SAR, the resident data acquisition computer system at Saturne, Saclay.

Finally, a triple beep accompanies the 'second alarm'. If this beeping is annoying, it can be toggled off (and back on) by "B" (=Beep).

All the above functions are followed by a statement to indicate the status of changes requested or made, for the purpose of confirmation. Also, the status of toggles are indicated on the menu list.

3.b.iv. Interrupts

If the inputted transmission is from an alarm, an alarm flag is set. Immediately after exiting the input subroutine, the screen routine goes to the interrupt routine. The transmission is then decoded, processed, and stored. There was not enough program memory to actually update the value seen on the screen with the alarm value, so upon exiting the interrupt routine, the screen mode is automatically switched to Alphameric mode. In refreshing the screen, the most current data is displayed, with exception to the

alarms that sent messages instead of a value. Also, the name, value (or message), time, and date were displayed at the bottom of the screen. This area, as previously mentioned is an area allotted to text, and does not change much with the screen modes.

There were three other types of interrupts: printout timer, system shutdown, SAFE Autostart alarm. Periodic printouts of pertinent variables were implemented by using a timer delayed interrupt in the SAFE 8000. The interrupt called routine in this case was an abbreviated form of the ordinary interrupt routine. The contents of the coded message, placed on the interrupt queue, was fixed: 1) "PP", 2) time, 3) date. After reception at the PC end, the transmission goes to the interrupt routine, is decoded, then branches to the printout routine. This routine prints out the name, voltage, and physical value of analog data, and name and on/off for digital. The reason for outputting the voltage is that the voltage was referred to more often for some data points. The format is fixed and somewhat customized.

The system shutdown message is actually a response from SAFE indicating that it received a shutdown command from Apple or has undergone an emergency shutdown, and has completed all shutdown operations except program termination. After the message is sent, SAFE shuts itself down, and the Apple decodes the message that shuts itself down: 1) "QQ", 2) time, 3) date.

While the PC shutdown is quite inconsequential, the SAFE must ensure that the acquisition system is cleared and, in an emergency shutdown, that the target is left in as safe a configuration as possible. This means that the target must be isolated and set to shutdown configuration. All SAFE controllable valves are closed in case the cell has begun leaking. Roughing pump #2 is left on. And to prevent explosive evaporation of the liquid H_2 , the cryogenic system is left on. It should be mentioned that the target isolation is effected by only two commands, not one device at a time.

The SAFE Autostart alarm informs Apple that while the Apple has been running, SAFE suffered a power loss and had started itself up again. This is typically a nonfatal

event in that the target usually is not running or is running on manual, and that the programs do not crash. The latter is possible, even if SAFE has been down for an extended period and Apple was hung up waiting for a response, because upon SAFE autostart a series of SAFE status lines are outputted. None of those lines, especially the first few characters, contain the letter "Y" which is the "received" code that initiates data input. So Apple ignores the incoming garbage and cycles to the next input which would be the SAFE Autostart alarm. No further action is taken.

There are two automated Apple program functions which interrupt the normal function of the Apple: memory transfer and disk dump. Both are regulated by the input subroutine in Apple, by means of counters. Since the graph mode can display 40 measurements in each graph, it was decided that when half that number was inputted the data should be shifted over. This would make room for more data but still leave some continuity in the graphs. The old data has to be stored so the data transfer routine is called to move this block of measurements from main memory to Auxiliary memory at Pages 96 to 122 (lines 370-690). This is done six times.

After the sixth memory transfer, a subroutine to transfer data to disk is called (lines 5240-5550). The data transfer routine transfers data, again in blocks of 20, from Auxiliary memory to the main memory. The memory locations containing the oldest 20 sets of measurement are used as a buffer, and oldest to newest blocks in Auxiliary memory are transferred. This way, the last block brought down is identical to what was written over at the beginning of the routine, and no resetting is necessary. Presently only one set of measurements out of ten are stored on disk. The target is mostly a static device and if there are any changes, they occur quite slowly, so that a frequent sampling would be overly redundant and a waste of disk space. A one in ten sampling is equivalent to two minute intervals (one cycle takes about twelve seconds).

The data files consist of 128 byte records. The format used in each record is as follows: 1) analog data #1;...16) analog data #16; 17) "-7" analog data terminator; 18)

digital data #1;...25) digital data #8; 26) "-8" digital data terminator. The first record of each data dump contains time and date only: 1) "-9" time record flag; 2) time; 3) date.

Alarms, if any, are also stored on disk, but since few alarms are anticipated, alarms are kept in main memory and dumped directly from there. Interrupt files have 32 byte records and are formatted as follows: 1) data type (0=digital, 1=analog); 2) data number; 3) data value; 4) time; 5) date.

Finally, the directory file is updated. The key concern is for the amount of space left on the disk. If the disk becomes full and a write is attempted, a DOS error results and the program halts. So the number of records is updated and the equivalent disk volume calculated. Normally, an Apple II disk can hold a little over 130K or 520 blocks (of 256 bytes each). A warning is displayed if 490 blocks are calculated to be occupied.

Both interrupts are followed by refreshing the graph screen. An empty graph is brought down from Auxiliary memory and refilled with the most current data. Then Apple returns to normal operation.

3.c. Start-up Program

Because of the shortage of BASIC RAM, the cold boot (i.e. starting at power up, as opposed to stop and start program) had to be broken up into a series of boot programs. The first one (H2 PC BOOT - see program listing E.3, Appendix E) resets the "Start of BASIC program" memory pointer to a value above Hi Res. Then it automatically loads the second program (H2 PC SET - see program listing E.4, Appendix E). This program creates a Hi Res character set (since no Hi Res character set was available) using Hi Res shapes¹¹, and stores it at Page 8. Next, the machine language (ML) data transfer routine is stored at Page 12. The Graph mode layout is then drawn up and transferred to Page 64 of Auxiliary memory from Hi Res by the ML transfer routine. Finally, it boots the main program. The only time these programs have to be used is if the computer has just been

powered up or memory has been corrupted.

At this point Apple defines and initializes its variables, then sends "R" to SAFE to query its execution status. If it is running, the SAFE communication interrupt routine sends back "X" and places a print dump interrupt on the output queue. If it is not running, SAFE will output garbage (actually a "syntax error" message) which does not contain the letter "X". In the former case, the "modify parameters" command is sent to SAFE. If the latter is the case then a **RUN 50** command is sent to SAFE to start it. Both cases are followed by Apple passing control to SAFE.

When SAFE is started, arrays and variables are defined and initialized, system timers are initiated, TASKs and interrupts are defined. Most of the variables are data point parameters: data point name, priority, timer delay, and set points. The name is important in identifying data points if they need changing. The interrupt priority has a maximum value of 23, but the alarm enable toggle is coded into bit-5 (=32 decimal). Hence, any value greater than 31 indicated the alarm for that data point is disabled. This is effected in the scan routines where the priority is checked before processing that data point any further. If the value is not satisfactory, the data point is skipped and the next TASK called. Furthermore, bit-6,7,8 are used to indicate a permanently disabled alarm. Since this represented a value greater than 63 and is not alterable during execution, these variables can not be accidentally toggled and are easily identifiable.

The timer delay refers to the interval given before an alarm is considered to be unrectified within a reasonable length of time. The values are expressed in 1/10ths of seconds and the default is 10 seconds (100 units).

Set points are values for data points which are considered alarm conditions. For digital alarms, they can only be on/off (coded 0/1 in SAFE). For analog, the set points are expressed as 12-bit digitized voltages. This is because the SAFE inputs the analog data as 12-bit information.

The program then goes to the 'main' routine, seven lines (lines 410-470) that

check a flag for commands sent from Apple, then TASKs. At start-up, this flag is set to "M" to execute the "modify parameters" routine. This is the same routine used for mid-execution parameter changes. A menu of three items is displayed: list parameters, change parameters, or go to main program. Listing parameters is actually a dump to screen of all data point parameters, defined or unused. Unused parameters are most readily identifiable by a variable name starting with a "D" or "A" suffixed with numbers.

The routine to change a parameter had a complication. INPUT could not be used to input the necessary information because it would interrupt data point scanning. Instead, the SAFE BASIC manual suggested using a GET (which can only input one character at a time) in a loop in series with a TASK.¹² The SAFE GET is an active one, in contrast to the Applesoft GET. If it is called and there is no input, it assigns a null character and passes execution to the next command (lines 3000-3030). Hence the loop continues indefinitely, calling TASKs (hence scanning) and concatenating each inputted character to a dummy variable until a terminating character, a carriage return in this case, is inputted. The name of the variable is inputted first. Then the program searches the name variable arrays for a matching name. Since digital and analog parameters are in different arrays, the data type is deducible here. A different menu displays the type of parameters that can be changed. The menu number and new value for the corresponding parameter are entered, upon which the appropriate routine makes the change.

When all changes are made, the enable status of all alarms are transmitted to Apple. This is somewhat vestigial; it was once planned to display alarm status with data. Since only the Raw Data mode has enough space for this, it is fairly useless information. On the other hand, at least the presence of this information in one screen mode meant for diagnostics would save the trouble of hitting "M", disrupting normal SAFE operation, and time. So it was left in.

Control is passed back to Apple which sends a time set command, the same one used by the user command because the same routine is used. The communications

interrupt routine sets a flag which is checked at the scanning level. Then the time set routine is called. This is fairly straight forward except that INPUT statements were used instead of the GET loops. This was mainly because time changes were not really expected to be made for long periods or during runs. It was even less expected that the date would need altering, so to avoid the hassle of answering frequently useless prompts, the date change prompt was called only if the time needed changing. The reasoning here was that if the time was wrong, it was only because of a power failure, hence the date would be wrong also. Of course, this is not necessarily true, so in case only the date needed changing, a null entry for new time allows the modification without actually altering the time. Finally, program control is returned to Apple, which goes to the main program starting at the Alphameric screen mode.

3.d. Support Software

Besides acting as a dumb terminal for SAFE when programming it, the Apple system was also used to store the SAFE program on disk. SAFE had a battery backed up memory so that it could retain a program when powered down. But it did not have any other means of program storage. For this purpose, programs were written to store and reload SAFE programs.

"LISTMAKER" was used to store SAFE programs onto disk. A major problem is that the first character of a line is occasionally missing at 300 baud, and frequently missing at 1200 baud. So this program should always be run at 300 baud. A "LIST" command is sent to SAFE by LISTMAKER to start the dump. It is also possible to store part of a program by specifying a partial listing (instead of a full listing) which sends a "limited LIST" command to SAFE. Each line is read, character by character, into Apple memory. At the end of a line (i.e. at a carriage return), an XOFF is sent to SAFE to pause the dump then the line is reconstituted. The reconstitution routine has to compare

the current line number with the previous one to verify ascending line numbers. If there is a missing digit, it is replaced and retested. When the line is in satisfactory condition, it is stored in memory. XON is sent to SAFE to read the next line. This is done for 100 lines, after which those lines in memory are dumped onto disk as records in a sequential file. The name of the file can be user defined, so that any number of different SAFE files can be stored on a disk (within disk capacity). At the end, the number of inputted lines and number of unreconstitutable lines are displayed to assure a clean transfer.

"LOADER" is used to load SAFE with a program from the Apple disk. This function is not complicated, owing to the reliability of the SAFE RS232, and can be used at 1200 baud. The same line 'fix' routine from "LISTMAKER" is included for redundancy.

"PRINTTEXT" is a program for reading and printing out the contents of the SAFE BASIC program stored in the sequential files on disk. This is far more convenient, due to speed, than getting the file listing from SAFE at 300 baud. The latter can be done using "PRINTOUT" which is basically the same program as "LISTMAKER" except the output device is a printer instead of disk drive.

To aid in program development, SAFE had a built-in program editor. The most commonly used features were selective and global renumber, search for strings, scan and replace strings, line delete and automatic line numbering.

Table III-1. Location of Data Points on SAFE 8000 I/O Cards

1.1 Analog Input (slot ch# 28,29)

ach#	variable	ach#	variable
0	TH1 (Target Pressure)	8	TC2 (Thermocouple at flask)
1	TH2 (Target Temperature)	9	n/u
2	VR5 (Comparator)	10	GP2 (Pressure Gauge)
3	GP1	11	GP3
4	GP4	12	GP6
5	LSB (Level Sensor - Top)	13	GP7
6	LST (Level Sensor - Bottom)	14	GP5
7	TC1 (Thermocpl at Cold Head)	15	n/u

1.2 Digital Input (ch# 4-11)

var#	dch#	db#	variable	var#	dch#	db#	variable
0	4	0	RP1 Manual	32	8	0	VP13 Manual
1		1	Pump On	33		1	Open
2		2	RP2 Manual	34		2	Closed
3		3	Pump On	35		3	VP14 Open
4		4	COMPRESSOR Manual	36		4	Closed
5		5	On	37		5	VP15 Open
6		6	FRIGe Manual	38		6	Closed
7		7	On	39		7	VP16 Open
8	5	0	DIFF. Pump Manual	40	9	0	Closed
9		1	On	41		1	VP17 Manual
10		2	CUTout	42		2	Open
11		3	AIR Low	43		3	Closed
12		4	N ₂ Low	44		4	VC4 (VH1) Manual
13		5	HOOD Off	45		5	Open
14		6	GAS Monitor	46		6	Closed
15		7	GP7D Ok/Nok	47		7	VC7 (VP12) Manual
16	6	0	115V	48	10	0	Open
17		1	+6V	49		1	Closed
18		2	GP2D Ok/Nok	50		2	VC8 (VP11) Manual
19		3	GP1D Ok/Nok	51		3	Open
20		4	GP3D Ok/Nok	52		4	Closed
21		5	GP4D Ok/Nok	53		5	VC9 (VP10) Manual
22		6	GP5D Ok/Nok	54		6	Open
23		7	GP6D Ok/Nok	55		7	Closed
24	7	0	LSB (Level sensor-bot)	56	11	0	VR5D Local
25		1	LST (Level sensor-top)	57		1	TH1D Control
26		2	n/u	58		2	Fridge COOLing
27		3	n/u	59		3	ByPaSs
28		4	n/u	60		4	SAFE On
29		5	n/u	61		5	ALARM
30		6	n/u	62		6	n/u
31		7	n/u	63		7	n/u

Table III-2. Location of Control Points on Digital Output Card (slot ch# 22,23)

var#	dch#	db#	variable	var#	dch#	db#	variable
0	22	0	RP2	8	23	0	VP16
1		1	COMPRESSOR	9		1	VP17
2		2	REFRIGERATOR	10		2	VP11
3		3	DIFFUSION PUMP	11		3	ALRM
4		4	VP12	12		4	SAFE
5		5	n/u	13		5	n/u
6		6	VP10	14		6	n/u
7		7	VP13	15		7	n/u

Table III-3 List of Data Points by Priority Interrupt

prio#	dch#	db#	variables
0	0	0:	
1		1:	reserved for print dump timeout
2		2:	
3		3:	
4		4:	
5		5:	
6		6:	
7		7:	
8	1	0:	V14 O, V15 O, VH1 O, V12 O
9		1:	V16 O, V17 O, V10 O
10		2:	V13 O, V11 O
11		3:	COMP, FRIG
12		4:	AIR, N ₂ , HOOD
13		5:	V16 C, V17 C, VH1 C, V10 C
14		5:	V12 C, V11 C
15		7:	V13 C, V14 C, V15 C
16	2	0:	RP1, RP2, DIFF, CUT
17		1	LSB, LST
18		2:	
19		3:	GP6, GP7, GP5
20		4:	GP2, GP3, GP4
21		5:	GAS, TH1, TH2, GP1
22		6:	115V, 6V, SAFE, ALARM
23		7:	(emergency shut down)

Table III-4. Graph Screen Graduations

TH1	17.0 psia 16.0 15.5	TH2	23.0 K 22.0 21.0 20.0 19.0 18.0 16.0	VR5	--	GP1	1.0E-3 1.0E-4 1.0E-5 1.0E-6 1.0E-7
GP4	1.0E-3 psia 1.0E-4 1.0E-5 1.0E-6 1.0E-7	LSB	10.0 V 8.0 6.0 4.0 2.0 0.0	LST	10.0 V 8.0 6.0 4.0 2.0 0.0	TC1	10.0 V 8.0 6.0 4.0 2.0 0.0
TC2	10.0 V 8.0 6.0 4.0 2.0 0.0	N/U	--	GP2	10.0 mbar 1.0 0.1 1.0E-2 1.0E-3	GP3	10.0 mbar 1.0 0.1 1.0E-2 1.0E-3
GP6	10.0 mbar 1.0 0.1 1.0E-2 1.0E-3	GP7	10.0 mbar 1.0 0.1 1.0E-2 1.0E-3	GP5	10.0 mbar 1.0 0.1 1.0E-2 1.0E-3	N/U	--

References

¹O. M. Alifanov et. al., "Hardware and Software Structures for Processing Thermophysics Experiment Data," Journal of Engineering Physics, **45**, #5 (1983), 1223.

²All subsequent references on SAFE BASIC are from the SAFE BASIC Syntax and User's Manual, version 1.05.

³C. A. Ludemann, "A Microprocessor Multi-Task Monitor," IEEE Transactions in Nuclear Science, **30**, #5 (1983), 3858-3859.

⁴Lon Poole, Apple II User's Guide, (2nd ed. Osborne/McGraw-Hill, Berkley, CA, 1983), p. 389.

⁵Apple II Extended 80-Column Text Card Supplement Manual, (1985), p. 15.

⁶*ibid.*, p. 29.

⁷John Collins et. al., "The IUCF Inter-computer Communication System LINKER," IEEE Transactions in Nuclear Science, **28**, #5 (1981), 3684-3685.

⁸E. Busse, et. al., "Data Acquisition and Monitor System HOOPSY at the Heavy Ion Accelerator VICKSI," IEEE Transactions in Nuclear Science, **28**, #5 (1981), 3675.

⁹University of California Biomedical Department statistics program. The reference for this program is W. J. Dixon et al., eds. BMDP Statistical Software 1981, (Berkley, CA: University of California Press, 1981).

¹⁰Apple II Reference Manual (1981), p. 6.

¹¹Poole, pp. 259-276.

¹²SAFE BASIC Syntax and User's Manual, 14-2.

IV. Asymmetry Analysis

The final contribution to the AHEAD polarimeter was a vector polarization analysis. The runs at Saclay, during the summer of 1986, were meant to calibrate the polarimeter for tensor polarizations. Since Saturne was not set up to produce one of these polarizations, it was necessary to rotate the polarization axis of symmetry. This was to be done by a solenoid and a dipole magnet. The solenoid was to rotate the polarization preferably onto the plane of curvature of the dipole, actually a spectrometer analyzing magnet, which then precessed the polarization so that it was not perpendicular to the beam. "Preferably" because rotating onto the bending plane of the dipole is a $\pi/2$ rotation, and would simplify later calculations of the required known tensor polarizations of the incident beam. But it was necessary to calibrate solenoid current to asymmetry (an indicator of polarization axis' direction). This was somewhat crudely done on-line, yielding a null Left-Right (L-R) asymmetry at a solenoid current of $I_0=183$ A. The work described in this thesis is a more careful analysis of the asymmetry dependence on solenoid current to verify that this was the correct solenoid setting.

1. Theory¹

For incident deuterons on an analyzer (such as liquid H_2), the differential cross section can be described in Cartesian coordinates by²

$$\sigma(\theta) = \sigma_0(\theta) \left[1 + \frac{3}{2}p_y A_y(\theta) + \frac{2}{3}p_{xz} A_{xz} + \frac{1}{6}(p_{xx}-p_{yy})(A_{xx}(\theta)-A_{yy}(\theta)) + \frac{1}{2}p_{zz} A_{zz}(\theta) \right] \quad (IV-1)$$

or equivalently, in the Madison convention (spherical tensor notation):

$$\sigma(\theta) = \sigma_0(\theta) \left[1 + 2\text{Re}(it_{11})\text{Re}(iT_{11}(\theta)) + t_{20}T_{20}(\theta) + 2\text{Re}(t_{21})\text{Re}(T_{21}(\theta)) + 2\text{Re}(t_{22})\text{Re}(T_{22}(\theta)) \right] \quad (IV-2)$$

where the p_i (or t_{ij}) are polarizations of the incoming beam and the A_i (or T_{ij}) are the

analyzing powers of the reaction. Polarizations are analogous to the spin state of the incoming particles in the beam and analyzing powers contain the physics of the reaction, particularly the effects of the potential of the target particle on the incoming particles (see Appendix A).

This is all in the reaction plane. To relate this back to the incoming spin symmetry frame, one considers angles ϕ , the angle the scattering plane makes with the incoming $x'-z'$ plane, and β , the angle that the axis of spin symmetry makes with the incoming z axis (see figure IV-1). ϕ can also be thought of as the azimuthal position of the detector of interest. The polarizations along the spin symmetry axis are called τ_{10} and τ_{20} (i.e. strictly vertical). As a result the polarizations become

$$\text{Re}(it_{11}) = \tau_{10} \sin\beta \cos\phi / (2)^{1/2} \quad (\text{IV-3a})$$

$$t_{20} = (1/2) \tau_{20} (3\cos^2\beta - 1) \quad (\text{IV-3b})$$

$$\text{Re}(t_{21}) = (3/2)^{1/2} \tau_{20} \sin\beta \cos\beta \sin\phi \quad (\text{IV-3c})$$

$$\text{Re}(t_{22}) = - (3/8)^{1/2} \tau_{20} \sin^2\beta \cos 2\phi \quad (\text{IV-3d})$$

The beam from Saturne comes out with $\beta=\pi/2$, i.e. vertically in the experimental frame. According to equations IV-3, this means that $\text{Re}(t_{21}) = 0$. But $\text{Re}(t_{21})$ is needed for the calibration (and experiment), so to create this polarization in the beam, a dipole magnet is needed, with its field perpendicular to the spin symmetry axis. This would cause precession of the spin axis perpendicular to the field, resulting in $\beta \neq \pi/2$. But the beam consisted of deuteron ions whose paths would bend in a dipole field. Since the polarization was vertically oriented, a beamline with vertical orientation would be needed, unless the initial polarization was rotated azimuthally before going into the dipole. This was accomplished using a solenoid. To simplify the relations resulting from such a rotation on equations IV-3, a rotation of $\phi_r = \pi/2$ was decided upon. But the solenoid current to ϕ_r relation was not known. To find the necessary current, Saturne produced solely vector polarized deuteron beams for several data acquisition runs. Equation IV-2

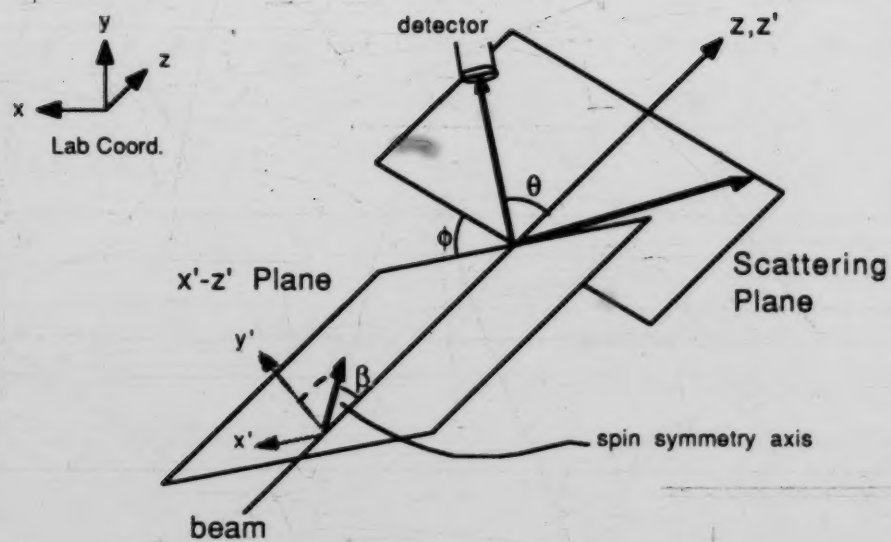


Figure IV-1: Coordinate Systems for Polarizations

then becomes the well known

$$\sigma(\theta, \phi) = \sigma_0(\theta) [1 + (2)^{1/2} \tau_{10} \operatorname{Re}(iT_{11}(\theta)) \sin\beta \cos\phi] \quad (\text{IV-4})$$

Left and Right are $\phi = 0$ and π , respectively, and the polarization is rotated by $\phi = \phi_r$.

$$L = \sigma_0(\theta) [1 + (2)^{1/2} \tau_{10} \operatorname{Re}(iT_{11}(\theta)) \sin\beta \cos\phi_r] \quad (\text{IV-5a})$$

$$R = \sigma_0(\theta) [1 - (2)^{1/2} \tau_{10} \operatorname{Re}(iT_{11}(\theta)) \sin\beta \cos\phi_r] \quad (\text{IV-5b})$$

Then:

$$(2)^{1/2} \tau_{10} \operatorname{Re}(iT_{11}(\theta)) \sin\beta \cos\phi_r = \frac{L - R}{L + R} \quad (\text{IV-6})$$

There are still four unknowns, but τ_{10} is fixed by the machine, β depends only on the dipole magnet field strength and the beam energy which were kept constant during the vector polarization runs, and $\operatorname{Re}(iT_{11}(\theta))$ can be kept constant by analyzing data from a fixed angular range. Hence the ϕ_r dependence can be found by a simple L-R asymmetry.

Equation IV-6 of course assumes an ideal experiment. In reality, there are bound to be instrumental asymmetries which can systematically shift all polarizations and, in the case of this analysis, result in a false current, I_0 . The main sources of concern are detector efficiencies, beam centering and consistent beam. With these in mind, a means of calculating the asymmetry can be derived. We begin by defining

$$(2)^{1/2} \tau_{10} \operatorname{Re}(iT_{11}(\theta)) \sin\beta \cos\phi_r = R_v \text{ and}$$

$$\text{pol up } L_{\uparrow} = K_{\uparrow} \alpha_L (1 + R_{v\uparrow}) \quad R_{\uparrow} = K_{\uparrow} \alpha_R (1 - R_{v\uparrow}) \quad (\text{IV-7})$$

$$\text{unpol } L_0 = K_0 \alpha_L \quad R_0 = K_0 \alpha_R$$

$$\text{pol dn } L_{\downarrow} = K_{\downarrow} \alpha_L (1 - R_{v\downarrow}) \quad R_{\downarrow} = K_{\downarrow} \alpha_R (1 + R_{v\downarrow})$$

"pol up" refers to the direction of the beam polarization, assuming that the accelerator has been set up for polarized beams. "Up" is the chosen "preferred" direction of the polarization. This is relevant when the polarization is alternately flipped or turned off by the accelerator. K refers to modifying factors common to a polarization state e.g. σ_0 , incident current. α refers to modifying factors common to each detector, i.e. efficiency. These two factors contain the sources of false asymmetries mentioned. It is then possible

to remove these faults by

$$\epsilon_{\uparrow} = \frac{L_{\uparrow} R_0}{R_{\uparrow} L_0} = \frac{1 + R_{v\uparrow}}{1 - R_{v\uparrow}} \quad (\text{IV-8a})$$

resulting in

$$R_{v\uparrow} = \frac{\epsilon_{\uparrow} - 1}{\epsilon_{\uparrow} + 1} \quad (\text{IV-8b})$$

Similarly

$$\epsilon_{\downarrow} = \frac{R_{\downarrow} L_0}{L_{\downarrow} R_0} = \frac{1 + R_{v\downarrow}}{1 - R_{v\downarrow}} \quad (\text{IV-9a})$$

resulting in

$$R_{v\downarrow} = \frac{\epsilon_{\downarrow} - 1}{\epsilon_{\downarrow} + 1} \quad (\text{IV-9b})$$

Equations IV-8b and 9b are usable if an unpolarized beam measurement has been done. But in the case of the calibration runs at Saturne, there were no unpolarized beam runs. The beam did flip polarization at every burst, though. To eliminate K and α we use

$$\epsilon^2 = \frac{L_{\uparrow} R_{\downarrow}}{L_{\downarrow} R_{\uparrow}} = \frac{(1 + R_{v\uparrow})(1 + R_{v\downarrow})}{(1 - R_{v\downarrow})(1 - R_{v\uparrow})} \quad (\text{IV-10})$$

$$\text{Define } R_v = (1/2)(R_{v\uparrow} + R_{v\downarrow}) \quad (\text{IV-11a})$$

$$\delta = (1/2)(R_{v\uparrow} - R_{v\downarrow}) \quad (\text{IV-11b})$$

and substitute into equation IV-10

$$\epsilon^2 = \frac{1 + 2R_v + R_v^2 - \delta^2}{1 - 2R_v + R_v^2 - \delta^2} \quad (\text{IV-12a})$$

Solve for R_v

$$R_v = \frac{\epsilon^2 + 1}{\epsilon^2 - 1} \pm \frac{[4\epsilon^2 + \delta^2(\epsilon^4 - 2\epsilon^2 + 1)]^{1/2}}{\epsilon^2 - 1} \quad (\text{IV-12b})$$

Rejecting '+' sign solution to maintain proper sign convention and using a Taylor expansion gives

$$\begin{aligned}
 R_v &= \frac{\epsilon^2 + 1}{\epsilon^2 - 1} - \frac{2\epsilon}{\epsilon^2 - 1} - \frac{(\epsilon^2 - 1)\delta^2}{4\epsilon} + \frac{(\epsilon^2 - 1)\delta^4}{32\epsilon^3} - \dots \\
 &= \frac{(\epsilon - 1)}{(\epsilon + 1)} \left[1 - (\epsilon + 1)^2 \left[\frac{\delta^2}{4\epsilon} - \frac{\delta^4}{32\epsilon^3} + \dots \right] \right]
 \end{aligned} \tag{IV-13}$$

If δ is small, then $R_v = \frac{\epsilon - 1}{\epsilon + 1}$ (IV-14)

The uncertainty, to begin with, is δ . Simultaneously, it is the remainder of the above series.

Solving for δ and folding in the statistical uncertainty:

$$\Delta R_v = \frac{4\epsilon}{N^{1/2}(\epsilon + 1)^2} = \frac{1 - R_v^2}{N^{1/2}} \tag{IV-15}$$

where N = total counts from both detectors.

As it turns out, as long as $N \gg 1$ and $\delta < 0.1$, this approximation is very good.

2. The Analysis

Now that the asymmetry can be expressed in terms of counts in L and R detectors, it becomes a matter of acquiring those counts. During the vector polarization calibration runs, only E counters E5, E6, E14, and E15 were in the trigger, along with the front end counters (see figures I-3 and I-5). This made AHEAD effectively a L-R detector system (instead of the large acceptance system it really is). Because the vector analyzing power iT_{11} is large at far forward lab angles (figure IV-2), the detector array was moved upstream 30 cm from its normal position (see figure I-3). This gave AHEAD more forward angle acceptance. Measurements were taken for 170 MeV incident deuterons with polarization flipped for each beam pulse, with solenoid currents stepped from 0 to 240 A. Typically, one run filled half a tape with a few hundred thousand raw events. Of that, pulsers accounted for ~2% and front end events for ~38%, leaving ~60% possible real events (see figure IV-3a). Of that 60%, approximately 30% lacked ΔE information,

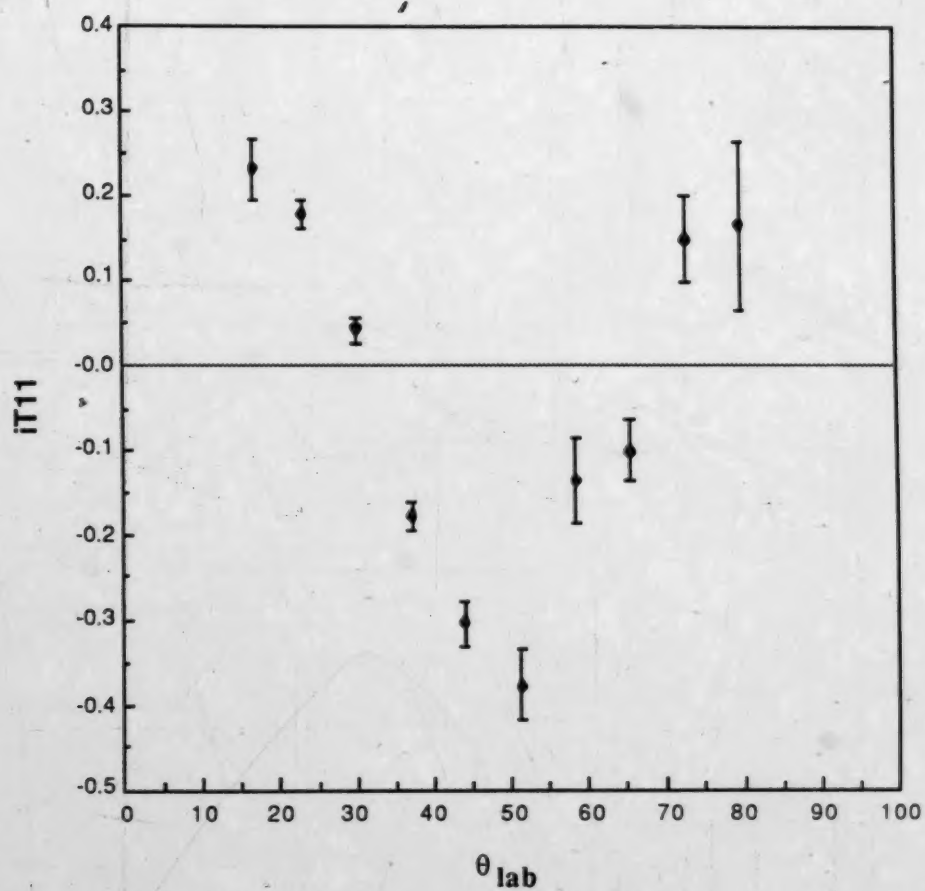


Figure IV-2: iT_{11} vs. Lab Angle for d-p elastic scattering at 191 MeV.⁵

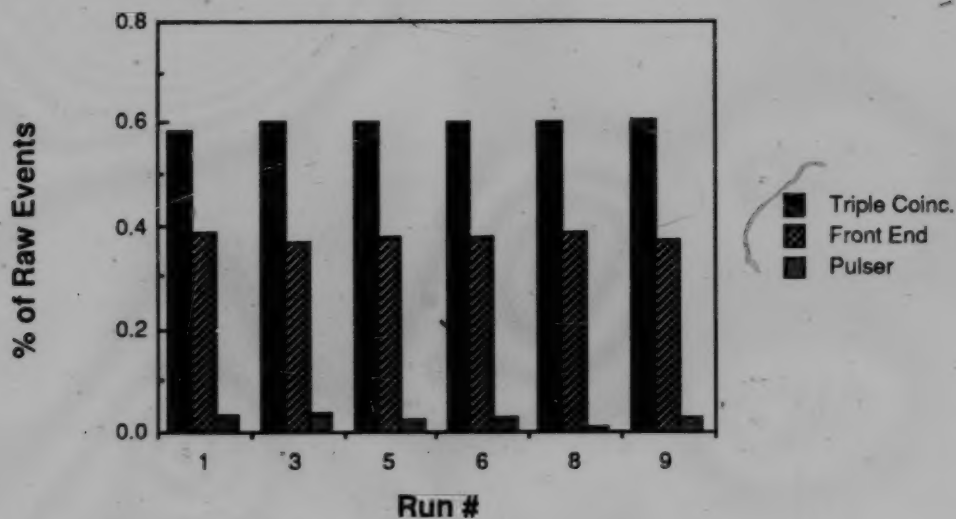


Figure IV-3a: Raw Event Breakdown, Q cut 0-15 cm runs

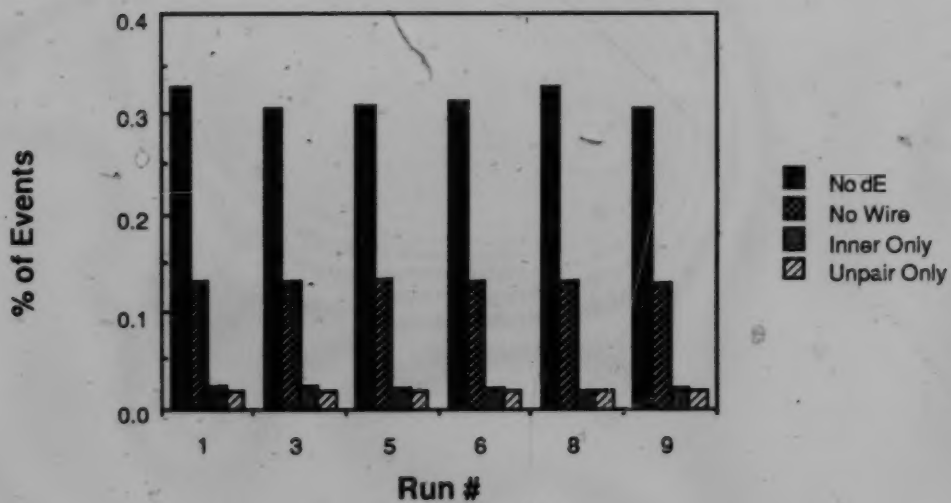


Figure IV-3b: Bad Event Breakdown for Triple Coincidence Events, Q cut 0-15 cm runs

~13% lacked wire information completely, ~2% had inner wire information only, and ~2% had outer wire information but no pairing was possible (see figure IV-3b). This left a total of about 30% possibly good events.

The first step in the analysis was to identify elastically scattered deuterons. Protons were not used because of the multitude of scattering channels possible. Deuterons would be limited to a very few channels, of which the desired elastic scattering is predominant at 170 MeV. Since 170 MeV deuterons that scatter forward and backward in the center of mass frame (CM) scatter in forward angles in the lab frame, with a maximum angle at 30° (figure IV-4), only the first 20° of AHEAD data was used. This was done by decoding outer wire chamber data and dropping any events that hit outside the requisite region, lengthwise for the far end of AHEAD. Track reconstruction was not performed because it was not necessary. Only relative changes from run to run were being studied, not angular distributions. Of course, the θ equivalence of these regions (or Q-cuts, as they were called) was necessary information, to avoid mixing backscattered and forward scattered deuterons near 30° (see figure IV-5). Also since the energies of forward and back scattered deuterons are significantly different at forward angles, they are easily separated (figure IV-4).

Having done that, the data were screened for events not striking ΔE_2 , ΔE_3 , ΔE_5 , or ΔE_6 , the ΔE counters associated with the E5, E6, E14, and E15, respectively. Since a stray event can go through more than 2 E counters, an event in one E counter was checked against a simultaneous event in its nearest neighbors. Passing that, it is possible that some events may be false ones, striking wires that do not correspond to the appropriate E counter. These were generally large angle scatterings that evaded the Q cut, since these were not previously screened (the result of these considerations is shown in figure IV-6).

Next was the time-of-flight (TOF) cut. It was initially thought that there was no TOF data on tape because of a mix-up in data acquisition. As it turned out, only 1 of 10

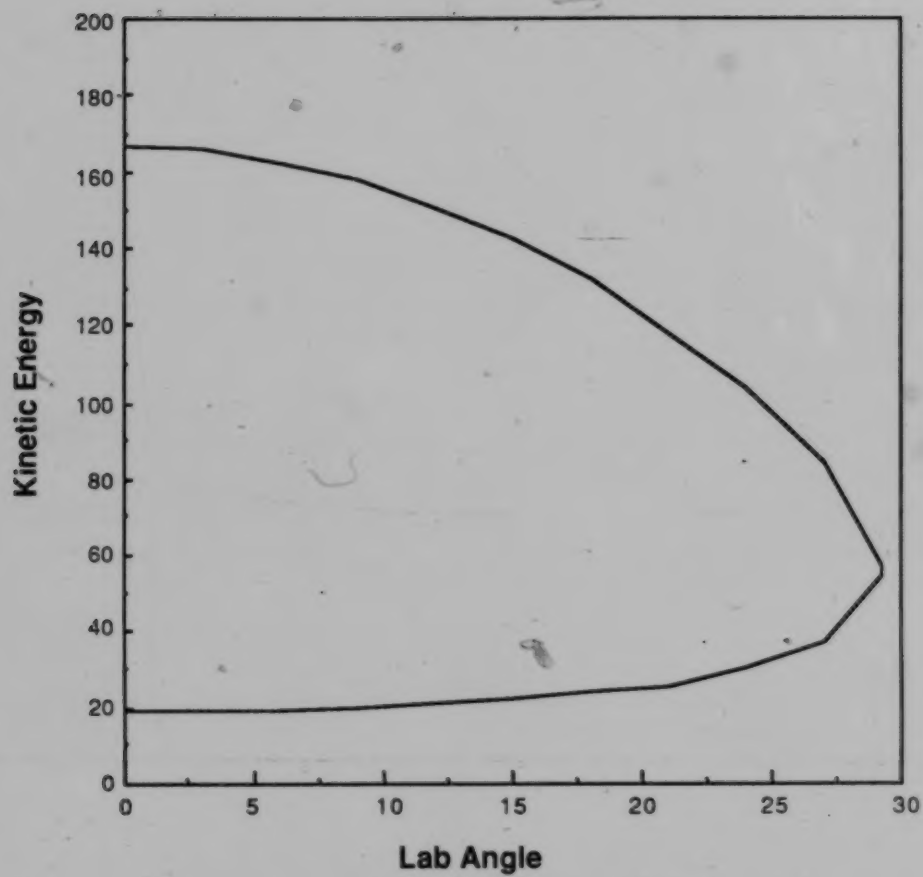


Figure IV-4: Kinetic Energy vs. Lab Angle
of Elastically Scattered 170 MeV Deuteron

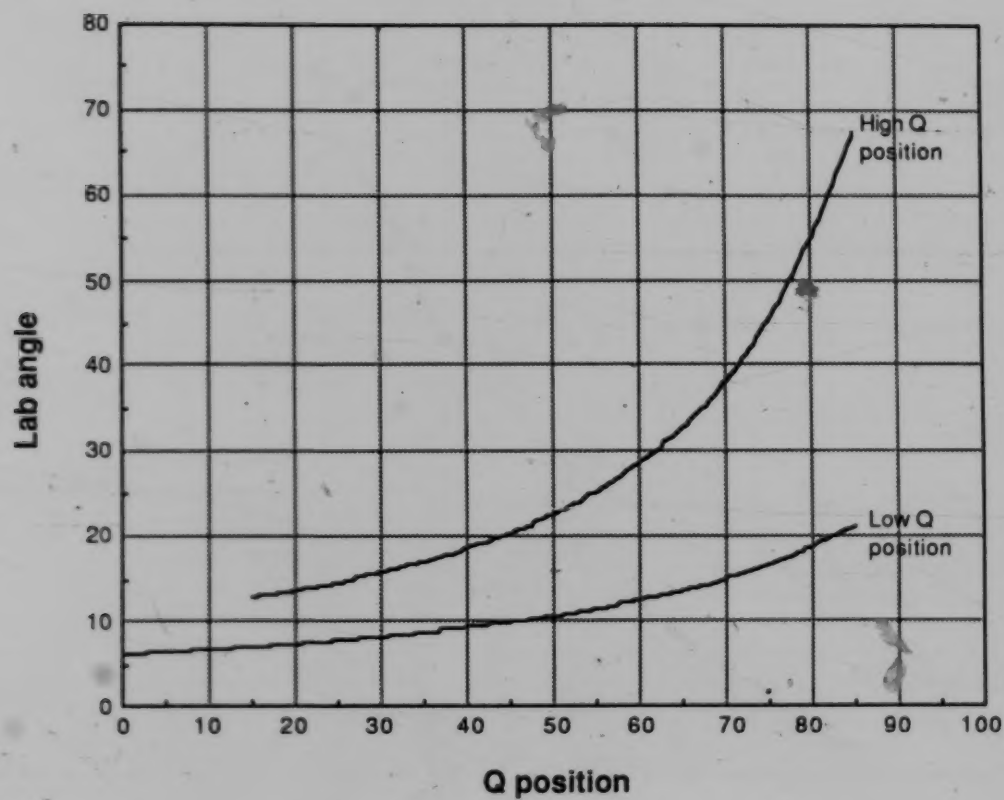


Figure IV-5: Lab Angle Envelope for Q-cuts. Q position is measured from the downstream end of the outer wire chamber.

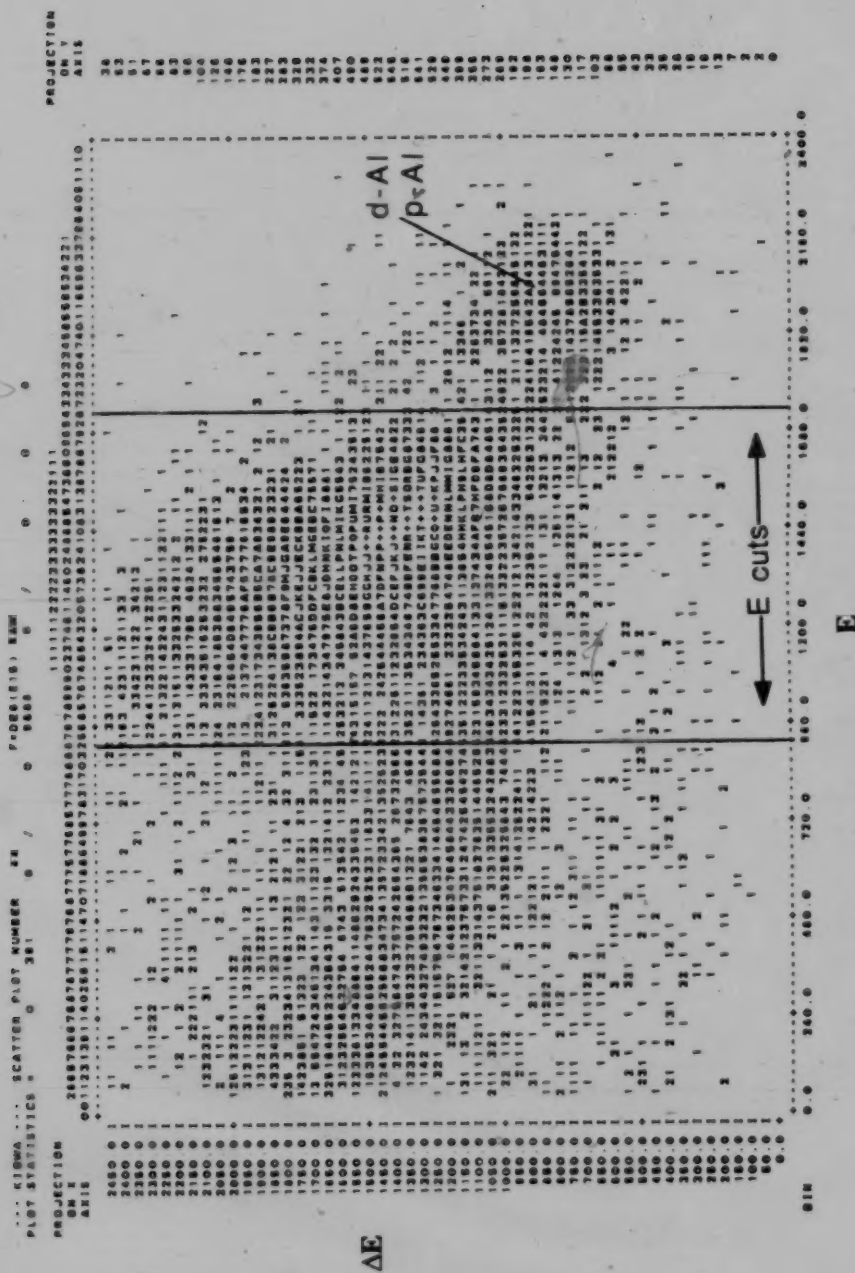


Figure IV-6: Raw ΔE vs. E plot for $\Delta E6$ and $E15$. 170 MeV d-p elastic scattering, 98.5 A Solenoid current, Q-cut Of 15-30, arbitrary scale.

vector polarization runs suffered from this (unfortunately, it was for R_v nearest to I_0). Deuterons are slower than protons at the same energy, and at 170 MeV, elastic deuteron TOF and proton TOF are noticeably different. So a linear TOF vs. E cut is made between the two (see figure IV-7). By the same reason, protons have lower ΔE than deuterons, so that the TOF cut eliminates lower ΔE events (compare figures IV-6 and -8). The actual dependence of TOF with E is non linear and the differences between the two kinds of TOF is small at low energies. But there should be no low energy deuteron events because, in principle, elastically backscattered deuterons would only have 20 MeV and would be absorbed in the ΔE counters. The net effect was that protons contaminated the low energy portion of the TOF cut. There were also what were believed to be elastically scattered d-Al events at high energies. Being deuterons, the TOF cut could not eliminate them. But they were quite separable with respect to energy.

So an energy cut was used. ΔE vs. E for elastically scattered deuterons at 170 MeV should be fairly localized for the small angular ranges used in this analysis. But due to the angular variations, small angles and the length of the target, the ΔE and E data tended to be spread out. For example, for a Q cut of 0-15 cm, $\theta \sim 6-13^\circ$. This corresponded to equivalent ΔE thicknesses $\sim 1.4-2.7$ cm (from 0.32cm), equivalent to 15-30 MeV energy loss for a deuteron entering with 150 MeV. The length of the H_2 target is about 30 cm, an energy loss of 25 MeV. The aluminium vacuum containment vessel (1 mm thick) represented a 10+ MeV loss, depending on exit point. This smearing made the E cuts difficult to place, mostly at the low energy side. It was eventually found that loose cuts and tight ones did not significantly change asymmetry results. So to eliminate the low energy contamination and the high energy d-Al events, a tight high energy cut and loose low energy cut were used throughout the analysis (see figure IV-8).

All cuts were unchanged from run to run to maintain consistency in results. Though there were variations in the width and centering of the ΔE vs. E distributions, the above cuts accommodated them without any deleterious effects. Consistent pulse data

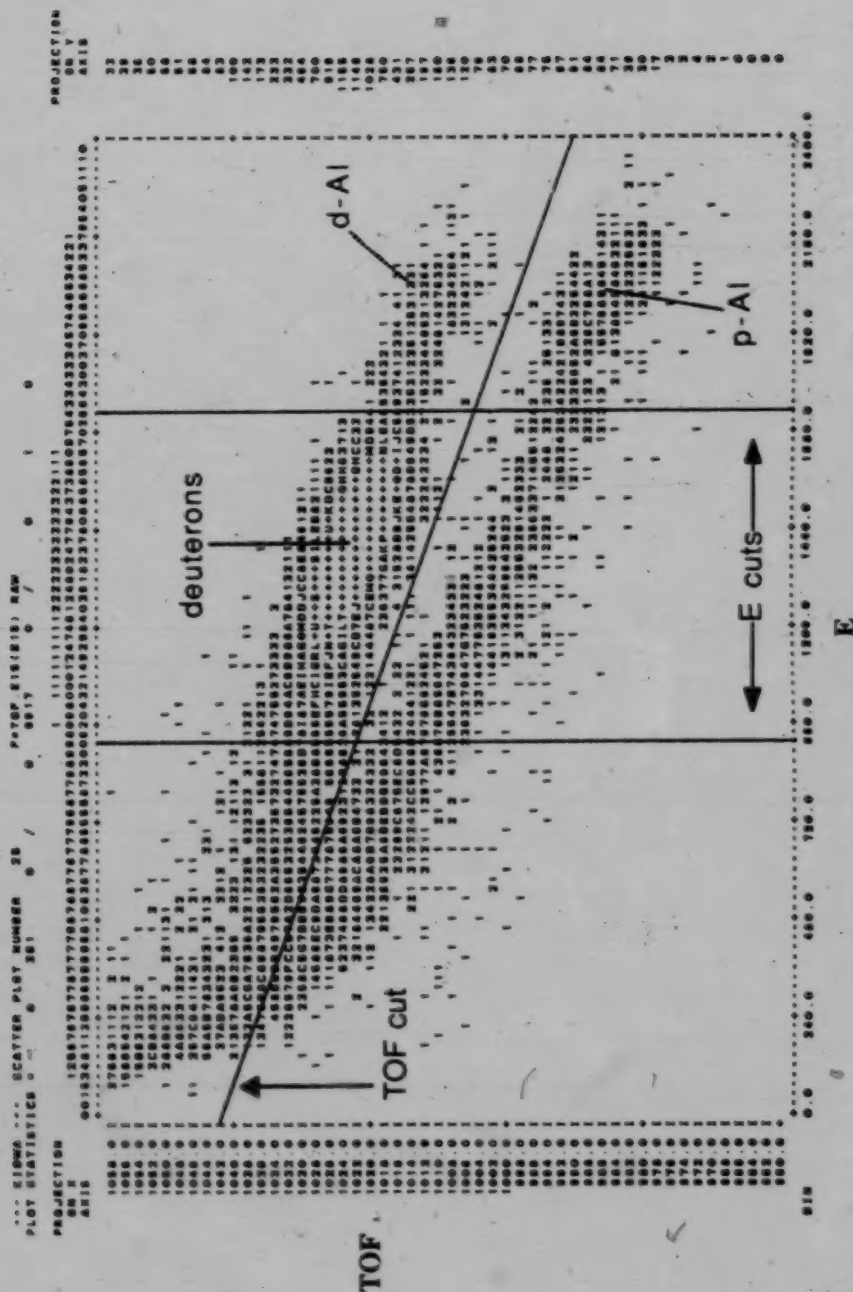
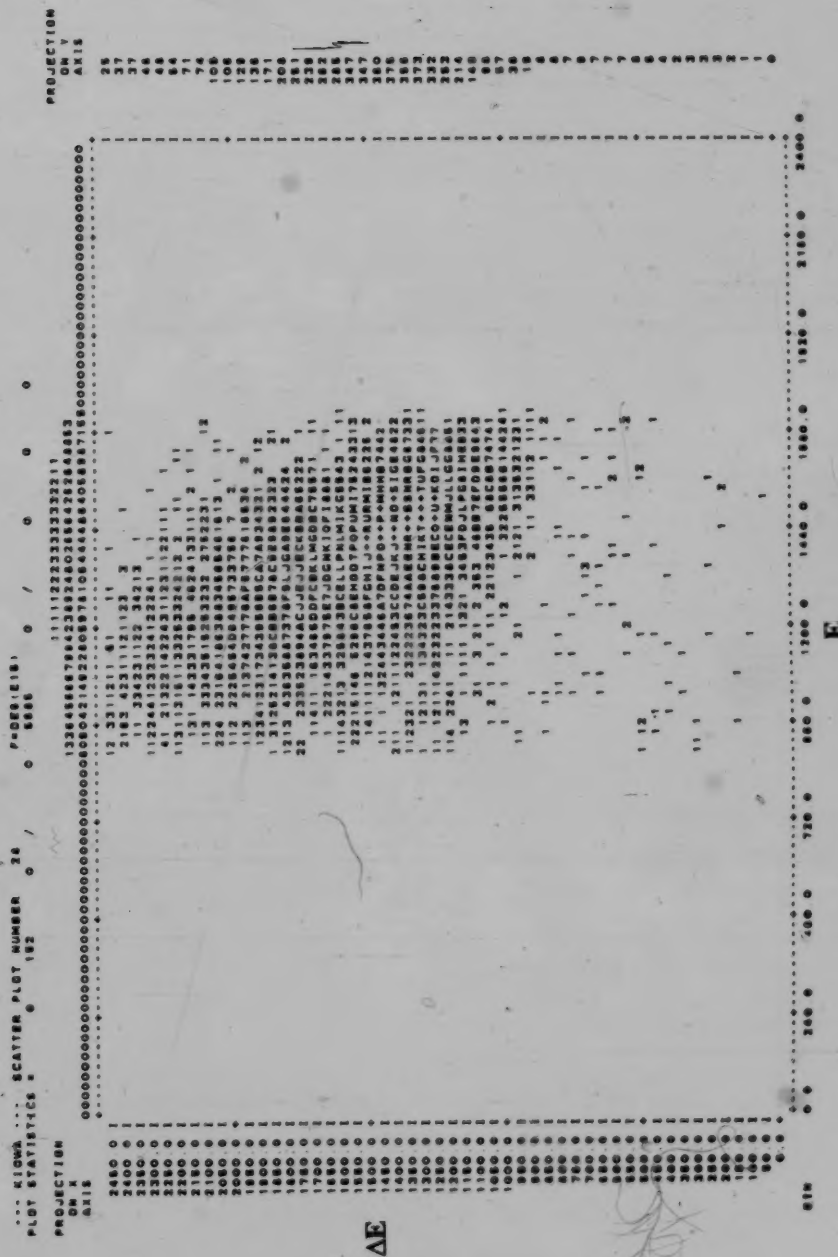


Figure IV-7: Raw Time-of-Flight vs. E for E15.
170 MeV d-p elastic scattering, 98.5 A Solenoid current,
Q-cut of 15-30 cm, arbitrary scale.



from some of the runs also support this.

After the data has passed all cuts, it can be used for asymmetry analysis. Since Saturne produced beams of alternating, known polarization, it was possible to separate events in each E counter into groups (or counters) for beam polarization up and polarization down. With this information, equations IV-10, IV-14, and IV-15 can be used to calculate the asymmetries. Assuming that the angle of rotation, ϕ_r , varies directly with solenoid current, R_v should vary with solenoid current sinusoidally. Hence a zero (or root) of a sinusoidal fit to R_v data should give I_0 , the null asymmetry solenoid current.

3. Program Details

The analysis was done on the University of Alberta MTS (Michigan Terminal System) Amdahl. The program used was KIOWA³ along with other pre-written code. Since the code was actually for another aspect of the AHEAD calibration analysis, it was necessary to modify and augment this version of the code. The INPUT routines were modified by a routine to read and store the polarization state as given by Saturne's control system. An extra subroutine, ALLDONE, was written for the purpose of calculating asymmetry at the end of an analysis replay from data stored by the main routine.

The analysis algorithm starts at INPUT. Here the data on tape was read and desired data decoded. There were four types of data blocks on the tape. The first block of every burst (scaler block) contained scalers for the data to follow. The polarization was the 18th and 19th word in this block. In principle, the scalers were supposed to be cleared at each burst. This was not the case during the vector polarization runs; they accumulated. Hence, some decoding was necessary before beam state could be stored in a COMMON block.

The second type were front-end events. This data is limited to the front end wire chambers and counters, and was ignored, save to count how many there were.

The third type were pulser events. It was necessary to ensure consistent operation of AHEAD, so pulsers for all detectors were fired at regular interval throughout a run. The replay of pulser data gave well defined peaks in the detectors' raw histograms, that did not change position. From this information, thresholds, gains, and pedestals were attained that were used in subsequent data. The results were consistent enough that one set of the latter data was used for all tapes. Pulser bursts shut out data from real events, so during normal analysis pulser bursts were also ignored.

The fourth type of data block was the real event, triple coincidence events. Not all data was decoded, just ΔE , E, ETOF, $\Delta ETOF$ and inner and outer wire chamber data. Inner wire data was not necessary, but its decoding routines were already in place and rather than spend the many hours to disentangle and remove it, it was left in. During decoding, if data was missing or undecodable, the appropriate counter was updated and the whole event rejected. The values of the counters were outputted at the end of the replay as event statistics shown in figure IV-3.

The wire chamber data had an extra complication. Wires were arranged to go from the front end of the chamber to the other, wrap around, and back to the front end, 3 positions over, in groups of 3 wires (see figure I-4). A good wire event needs both wire ends to fire, so an event at one wire end is checked with its other end. If there are not 2 hits, then the wire data, not the event, is rejected.

Wire data is ADC data representing charge received by converters. A charged particle ionizes some gas molecules in the chamber, and the ions deposit charge on a wire. This charge flows to the wire ends (and converters), but the charge amplitude is attenuated due to the resistance of the wire. This gives some idea of the distance that the charge had to travel; the further away an event is from the wire end, the lower the charge. Of course, as the distance from one end increases, the distance to the other end decreases. Hence, for a good event it is impossible for the sum of charge from both ends to have low output. If this does happen, which was fairly frequent, there has been a misfiring. The

check for this is a QSUM cut; if the sum of charges of the paired wire ends was less than a defined number (250 for this analysis) it was rejected.

If the event was successfully decoded, then a position cut was made. This is the previously mentioned position cut (Q-cut) to limit data to forward lab angle scattering. Then control is given back to the main program.

The main program, as far as this analysis was concerned, executed commands from a control file. It performed the rest of the analysis as outlined previously. A ΔE was identified, and the E associated with it. Then the nearest neighbors were checked. At this point, a user defined function, USR, was called. It checked the wire to E correspondence, and if successful, makes corrections to E and ΔE data due to photon pathlength attenuation (not to be confused with deuteron pathlength).

After exiting USR, raw ΔE vs. E and ETOF vs. E scatterplots were updated. Subsequent scatterplots could then be compared to them. Scatterplots and histograms were also defined in the control file.

TOF and E cuts were made. Counters for $L_{\uparrow}$, $R_{\uparrow}$,... and scatterplots for a few different test combinations were updated: TOF and ΔE vs. E scatterplots and counters for events succeeding and failing TOF and E cuts; counter for events passing E cuts with no TOF cuts, to compare with on-line results; counters for events passing all cuts but with ΔE too low. Of the latter, there were very few, but they showed polarization similar to the normal deuterons. It is suspected that these are protons from backscattered deuterons that snuck past the admittedly crude identification algorithm.

The above algorithm was followed one event at a time. Counters for failing and/or passing the various tests were incremented for each event. At the end of a replay, statistics from these counters, histograms, and scatterplots were outputted. KIOWA finally called subroutine ALLDONE to compute the asymmetry from the up and down counters of E5, E6, E14, and E15.

Data acquired for 6 different solenoid currents were used for the vector

polarization analysis. A sinusoidal function was then fit to the resulting final 6 R_v . The program used was the U of A MTS version of BMDP statistical package with a user defined function of $R_v = p_1 \sin(p_2 I + \pi/2)$. The function was originally $R_v = p_1 \sin(p_2 I + p_3) + p_4$, but for several reasons was reduced to a 2 parameter fit. p_4 was introduced due to a suspicion of the presence of a systematic R_v shift upon inspection of on-line and 0-15 cm position cut results. The 200 and 240 A data points seemed too low. While there are possible sources for systematic error, the idea was dropped when the 15-30 cm position cut results showed no such problem.

The phase shift was fixed at $-\pi/2$ because there were not enough data points to reasonably fit the function. Fits with a free phase shift tended to have unrealistically displaced maximum R_v , implying that the polarization coming out of the accelerator was not consistently vertical in the lab frame, or that the dipole was affecting the vertical component of polarization.

A sine function was used instead of cosine because the sine function had simpler derivatives (no negative signs to account for) and zero for a root, the former necessary for the BMDP function definition. The presence of negative signs and non-trivial roots could have caused some problems in debugging.

Three sets of asymmetries were calculated: a replay set with Q cut of 0-15 cm, a replay set of 15-30 cm, and a recalculated set from on-line data for comparison (Table IV-1, figure IV-9). Three fits were made and 3 I_0 calculated from the fits, shown on Table IV-2. The August estimate was very close to the value from a fit of the same data. The 0-15 cm data does not conform as well to a sine fit and this is reflected in the uncertainty of its current. The value from a Q-cut of 15-30 cm is the lowest of all, but its fit is good, its uncertainty is low, and it is within error of the calculated values. The weighted average of the results of this analysis is consistent with the on-line results. Hence the basic I_0 used during subsequent runs was probably adequate enough to assume that the spin axis of symmetry was rotated onto the bending plane of the dipole magnet.

Table IV-1: Asymmetries for different Solenoid currents and different analyses.

Run No.	Solenoid Current	On-line R_v (%)	R_v w/Q-cut of 0-15 cm (%)	R_v w/Q-cut of 15-30 cm (%)
1	0.0A	-13.469 ± 1.234	-15.780 ± 1.243	-15.540 ± 0.791
3	47.5A	-11.973 ± 1.032	-16.170 ± 1.344	-13.240 ± 0.845
5	98.5A	-9.007 ± 1.078	-10.162 ± 1.016	-10.230 ± 0.644
6	148.8A	-4.588 ± 1.134	-3.805 ± 1.891	-4.452 ± 1.220
8	199.1A	2.812 ± 1.013	3.501 ± 1.876	4.376 ± 1.181
9	240.5A	6.182 ± 1.065	4.500 ± 1.706	7.457 ± 1.076

Table IV-2: Null Asymmetry Solenoid Current and Parameters from sine fits from different analyses of vector polarization data.

Analysis	Fit Parameters		I_0
	m_1 (%)	m_2 ($10^{-3}/A$)	
Aug '86			183 A
Recalc On-line	13.41 ± 0.41	8.640 ± 0.141	181.8 ± 3.5 A
0 - 15 cm	15.84 ± 0.88	8.399 ± 0.366	187.0 ± 8.1 A
15 - 30 cm	15.31 ± 0.45	8.839 ± 0.198	177.7 ± 4.0 A
Weighted average of this analysis			179.5 ± 3.6 A

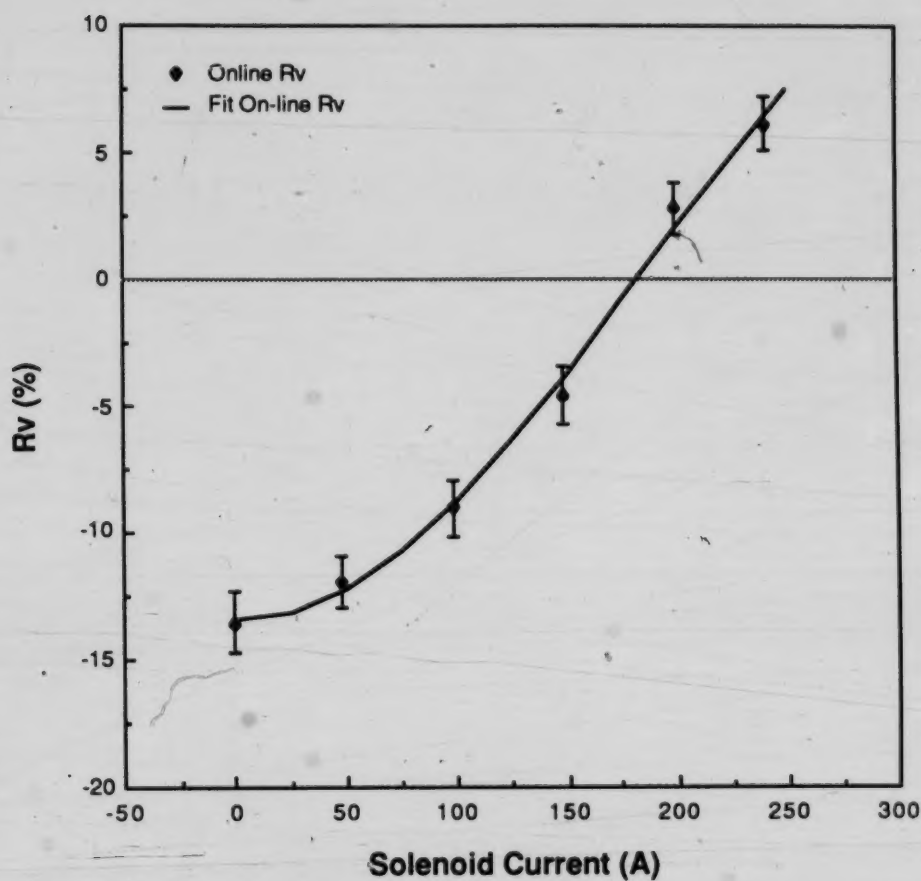


Figure IV-9a: On-line Asymmetry (R_v) vs. Solenoid Current
d-p elastic scattering at 170 MeV. Fit curve is a BMDP fit.

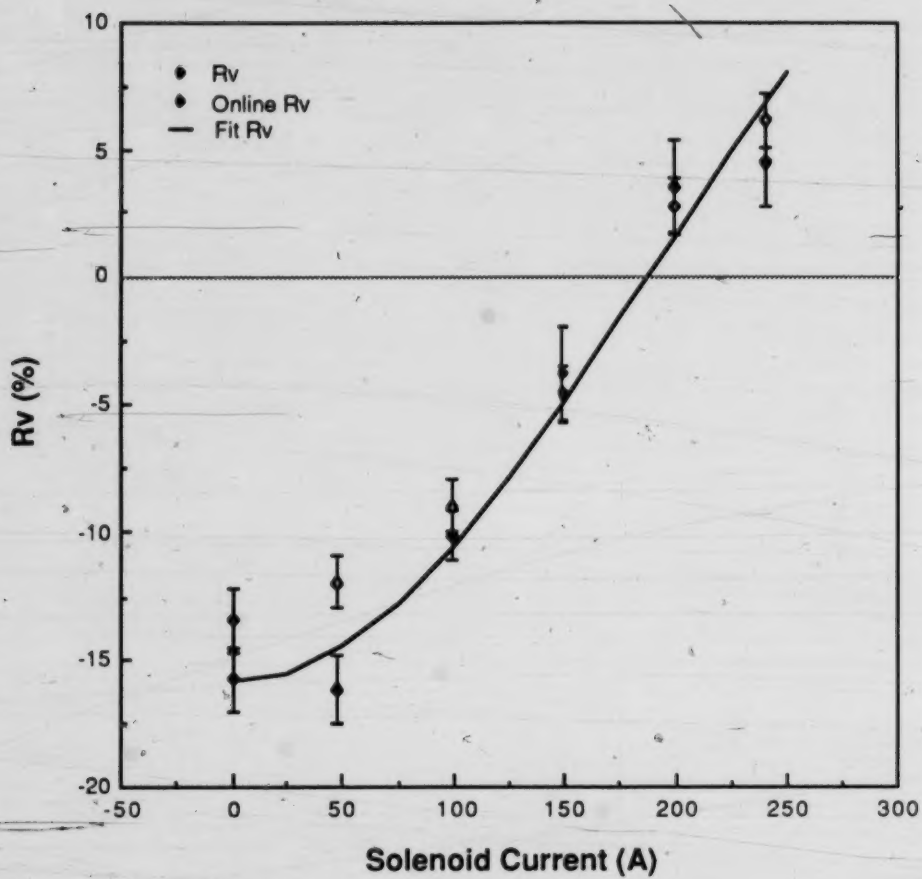


Figure IV-9b: Asymmetry (R_v) vs. Solenoid Current
d-p elastic scattering at 170 MeV, Q_{cm} of 0-15 cm.

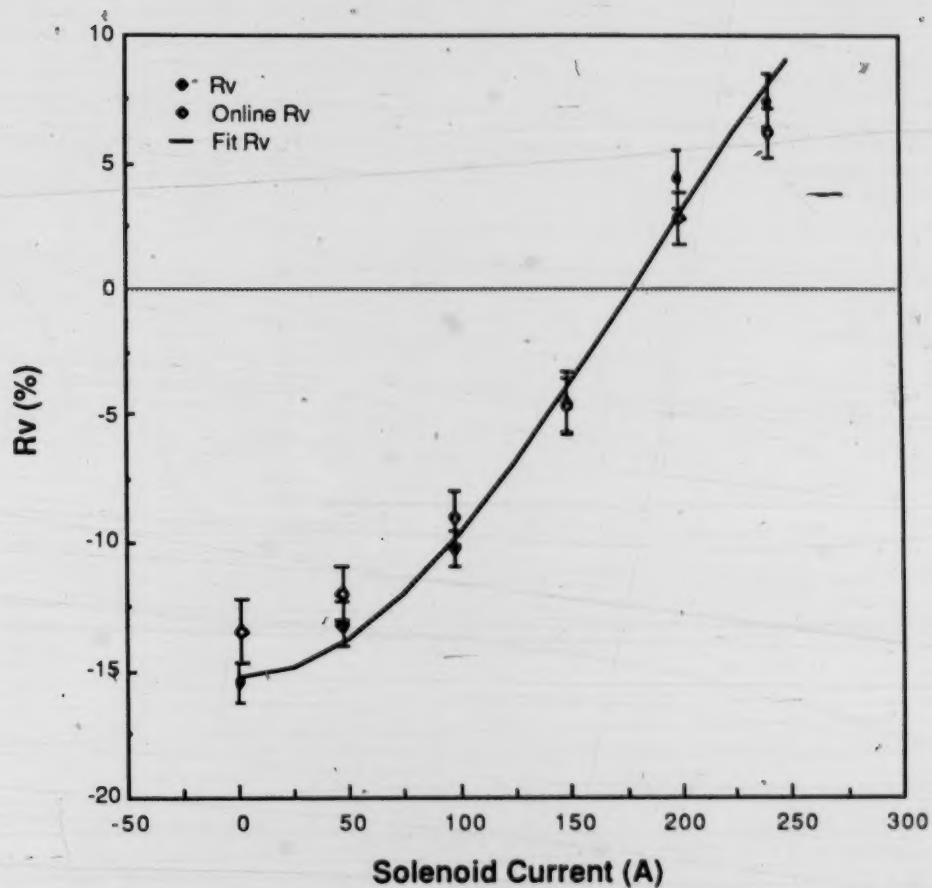


Figure IV-9c: Asymmetry (R_v) vs. Solenoid Current
d-p elastic scattering at 170 MeV, Q-cut of 15-30 cm.

References

¹All references in this section, except where noted, are from W. Haeblerli, "Polarization Experiments" in Nuclear Spectroscopy and Reactions, Part A (ed. J. Cerny, Academic Press, NY, 1974), pp. 151-191.

²G. Ohlsen, "Polarization Transfer and Spin Correlation Experiments in Nuclear Physics," Report on the Progress of Physics, **35** (1972), p. 737.

³Albert W. Stetz, "KIOWA - Data Histogramming and Plotting," University of Alberta, Nuclear Research Centre Internal Report #81.

⁴6 tape runs out of 10 were found to be usable for the vector polarization analysis, each at different solenoid currents. Run 2 was an aborted run. The tape holding Run 4 and 5 was physically damaged during shipping. Run 5 survived, but Run 4 could not be used. Run 7 did not contain any data. Run 10 did not have any TOF information. So that left Runs 1, 3, 5, 6, 8, and 9 that were almost all usable.

The exceptions were Runs 6 and 8. It seems that during some of the calibration runs, the operators prematurely changed the solenoid current. Hence, part of some tape runs were garbage. The extent of this problem was found by replaying parts of a run (usually half, which was sometimes followed by 3/4) and comparing results from the different replays for the same tape run. For Runs 6 and 8, the last half of the run showed asymmetries greater and smaller, respectively, than the correct asymmetry by about 40%. This analysis was performed for data with a Q cut of 15-30 cm. The 0-15 cm data did not have enough statistics to judge the significance of any differences.

⁵M. Garcon, et. al., "Measurement of Vector and Tensor Analyzing Powers for 191 and 395 MeV Deuteron Scattering," Nuclear Physics A458 (1986), p. 294.

V. Monitor and Target Performance

1. Target Cryogenic Performance

There were essentially three conditions in which the AHEAD target functioned: cool down, normal run, warm up. Of the three, the cooldown is of the most interest, since one would like to know when the target will be ready to run. The vacuum system is mostly static. It is the cryogenics that is dynamic and so is important.

There are two stages in this process. H_2 gas in the condenser and flask must be cooled to liquefaction temperature, then the flask fills. The pressure is held relatively constant and H_2 gas allowed to enter freely. Temperature in the condenser and at the top (LST) and bottom (LSB) level sensors are monitored. The condenser temperature comes from TH2, a vapor pressure bulb. The vapor pressure was found to be linear with temperature, but the pressure varied as a rational power of voltage. The result is something resembling an exponentially decreasing T vs. t curve. At the liquefaction temperature, T_1 , TH2 plateaus.

The level sensors were resistors whose voltages changed with temperature. They were originally meant to register liquid at their positions when their voltages passed their set points. But as seen in figure V-1, the status of cooldown can be inferred from the level sensors, as well as from TH2. LSB will cool almost in unison with TH2 since contact between the condenser and flask is through the bottom feeding tube. Hence it plateaus with TH2. Assuming a simple reciprocal relation, LSB also shows an exponential relation and a plateau.

Since the flask is an insulated container, except at the feeding tube, LST does not change as LSB or TH2. It lags (in temperature) but it does show a shallower exponential increase until liquefaction begins. At this point, instead of plateauing, it slowly linearly increases until the H_2 level reaches it, whereupon its voltage shoots straight up. If one

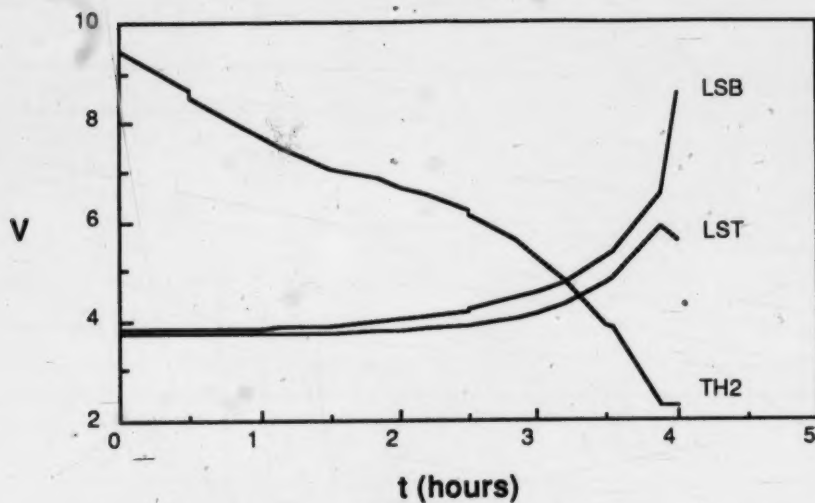


Figure V-1a: Target Sensors during Preliquefaction Cooldown of Target.
TH2 is the Vapour Pressure temperature sensor, LSB is the Bottom Level Sensor, and LST is the Top Level Sensor

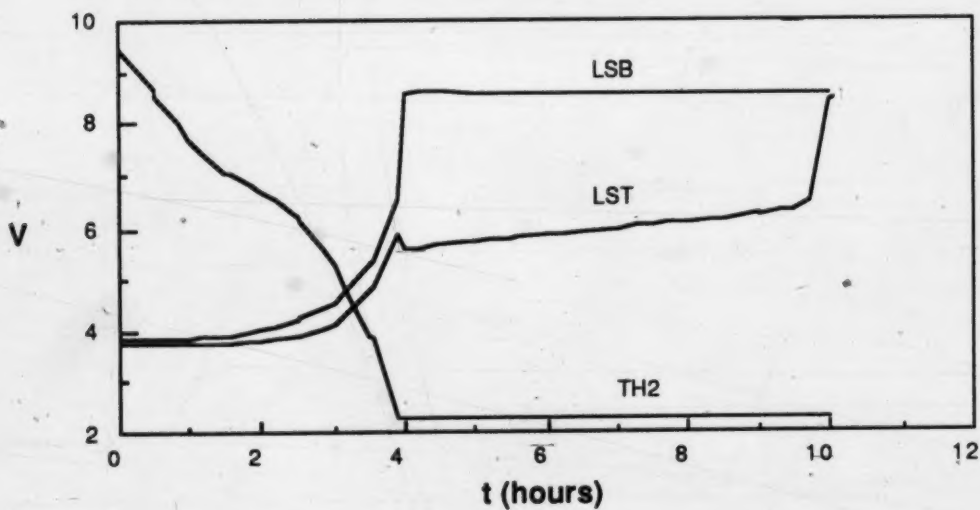


Figure V-1b: Target Sensors during Full Cooldown of Target.

assumes that LH_2 is supplied at a constant rate from the condenser, then LST can easily be used to indicate the volume of liquid and the level, during filling.

The data in figure V-1 is a compilation of 2 out of 4 different cooldown runs taken from 3 printouts and 2 disk files, taken on 4, 11, 18, 25 Aug., 1986. There was some difficulty in combining the data. The first step was to fit polynomials to each data set. For the cooldown, the highest orders used were 5 for TH2 and 4 for level sensors. Exponentials were not used simply because they did not give good fits and exponential series would be difficult to correlate. The latter objection became irrelevant when the polynomial series turned out to be difficult to correlate.

The main concern was to be able to compile a fairly accurate record of the target's time dependent thermodynamics so that future runs with the target can be compared with some standard. This required correlating the times of the data sets. For data recorded on the printer, this was thought to be simple enough, yet highly consistent results were not possible. Disk data was complicated by the lack of regular timing when recorded, so an additional unknown due to the uncertainty of the intervals was introduced. To make matters worse, disk data was not complete because at the time the disk routine was not completely developed.

When exponential fits failed to work on the preliquefaction data, the only alternative was to use polynomial fits. The fitting was simple and yielded exact values if the least squares method was used. A program which compiled the appropriate matrix elements and then row reduced the matrix was written on a Commodore 64. Then various fits were attempted; the order of the polynomial was altered, various data points were removed or replaced, weights were changed. End points were most subject to removal due to their proximity to discontinuities in thermodynamics. Weights were varied for the same reason. Fits were made on LSB, LST, and TH1 data for consistency of the correlation parameters between them.

At first, the fits were voltage vs. time, and a non linear correlation was made by

varying multiplicative and additive parameters for time. This was also a least squares type fit, with the parameters incremented according to their respective deviations (iterated increment "halving"). This was of course the hard way. It was then realized that a time vs. voltage fit would simplify matters. This way voltage, which does not suffer from a variable reference point, from any data set will yield time from a standardized frame. The calculated new time was then linearly compared with the original time for that set of voltages by a linear least squares method. The resulting parameters were time correlation parameters: the multiplicative one being a scaling factor more important for disk data, and the additive being the reference time shift.

Uncertainty was not a great concern because this analysis was mainly for approximate predictions. As long as the correlation parameters were consistent there was no problem in taking a weighted average of the parameters. Some parameters were completely incompatible, due mostly to a low degree of freedom of data, and were rejected. In some cases, the magnitudes varied substantially but their weights were low. With this information, the data from the three printouts could be combined into a more detailed data set. It should be noted that because AC power at Saclay was at 50Hz, the cryostat compressor will work faster at Bates, so that the time scale in figure V-1 should be compressed by a factor of 5/6 when running at Bates. This factor was noted during tests at Saclay when compared to previous cooldown tests at the U of A Nuclear Research Centre.¹



2. Possible improvements on Monitor

The monitor system was used in the August 1986 calibration runs without significant problems. The most commonly used facilities were the Alphameric screen and the printout. By cross-referencing with previous printouts it was possible to ascertain the status of the target system.

The biggest complaint voiced for this system was the speed. Ten to twelve seconds per cycle seemed interminable. The cause of this problem is from the data transmission and decoding, and computations (most of the Apple program). Upon reflection, it would seem that the "high level" rule was applied too firmly. Machine language (ML) subroutines at strategic locations (perhaps with labelling or documentation) would have increased speed significantly. One notes that the transfer of 8K of the Hi Res screen from Auxiliary to Main memory took a fraction of a second, compared to the ~1 second that a BASIC routine took to shift the 5K of the temporary data storage.

The data input was limited by some inadequacy of the Apple RS232 and by the character string buffer. An ML input routine would be fast enough to capture all incoming data flawlessly and to buffer that data efficiently - provided that the loss of first characters is not a hardware problem. Concerning the latter, it is curious that the built-in dumb terminal program also displays this problem, indicating a timing problem. The data loss problem, however, is rather minor for the monitoring aspect of the system (solved by padding the beginning of data lines with spaces). The data buffer would be essential. The use of the string data buffer costs the system 2-3 seconds due to clearing. If a fixed location was used, there would be little need to use the FRE function to clear the buffer. Data can then be reconstituted without resorting to string variables. A simple and fast clear routine could be used: simple because there would be no need to be selective and the area would be unchanged, fast because only a small amount of memory would be needed

(e.g., 256 or 512 bytes). This routine would replace the input loop containing the INPUT and the XON/XOFF commands.

It has been noted that digital data could be decoded more efficiently in ML. The suggestion is strongly urged here. Such a routine would cut cycle time by about 25%, and would reduce program size. It is doubtful that analog data can be further treated in ML. Doing so would duplicate the functions already existing in BASIC. The data in temporary storage in the 8K "dead space" could be shifted by ML much faster than the looped POKE-PEEK routine currently in the program. It is suggested, however, that the location of the affected memory be parameters alterable in BASIC because the temporary storage may increase if more data points are added to the system. This is how the Auxiliary-Main data transfer routine is directed.

Another idea strongly urged is to store all screen formats in Auxiliary memory, then transfer them from there whenever a screen needs refreshing. This method was tested with the Hi Res screen. The screen format was created and stored in the boot program. This reduced program size and increased speed. The "test" was to assure that the memory contents did not suffer corruption during a typical run due to software or hardware bugs. No problems occurred in this aspect, so it has been deemed safe practice.

Machine language routines can be called in two ways. The CALL X goes directly to the memory location X, where the routine is, and executes it. The disadvantage with CALL is that the function it executes is not recognizable. Only with documentation, on paper or in the program, would it be possible for its true function be apparent. Also, if the routine requires parameters from the program, then those parameters have to be POKEd into the appropriate memory locations before the CALL. If the routine is to compute some result or requires a single numerical parameter inputted or both, the USR(X) function may be more practical. It is treated in BASIC as another mathematical function, where X is the value or variable that is to be operated on, and when control is returned to BASIC, USR returns the computed value. Again, this suffers from a lack of

recognition. But it can be assigned to a user defined function (by the DEF FN command) which can be labelled, so that objection can be eliminated. Memory pointers also have to be set to direct USR to the desired ML routine, by POKEing, and if the ML routine requires two or more parameters, they also have to be POKEd into memory.²

There is a possibility that the vacuum gauge conversions can be done more quickly (and more accurately) using a polynomial fit (see figure V-2). The key would be to express powers in terms of iterated products instead of using the BASIC powers function, " $\wedge$ ". Multiplication is about ten times faster than the powers function so that the highest order of the polynomial fit should not exceed 3. In that light, line 1620 should be changed to

XP=DA(M)/409.5:MN=.7137*XP-2.7584-.0723*XP*XP+.0049*XP*XP*XP

and line 1650 should be changed to

XP=DA(M)/409.5:MN=.8761*XP-6.9928-.0727*XP*XP+.00337*XP*XP*XP

The corresponding lines in the printout routine (4740 and 4770) should also be changed.

There are some suggestions which are not related to speed. First, the digital alarms (e.g. for the valves) presently are indicated by flashing the data point on the screen. A beep could be attached to this by concatenating variable G\$ to the PRINT statements in question. This reduces the need for someone to check the screen intermittently. Another suggestion is to transmit time information to Apple with data and storing it with the data. The extra amount of data memory needed would be negligible compared to what is available (see figure III-1). This would keep the screen time more up to date and solve a small problem. Presently, the disk data suffers from a lack of definite and accurate timing. By including the time of transmission in stored data, a more accurate log can be kept on disk. This was not done earlier because of speed and memory considerations. But with the above changes, both considerations should be bearable.

A final note. There are two bugs in the SAFE program. The IF statement in line 660 skips over an identification statement. That IF statement in line 660 should be

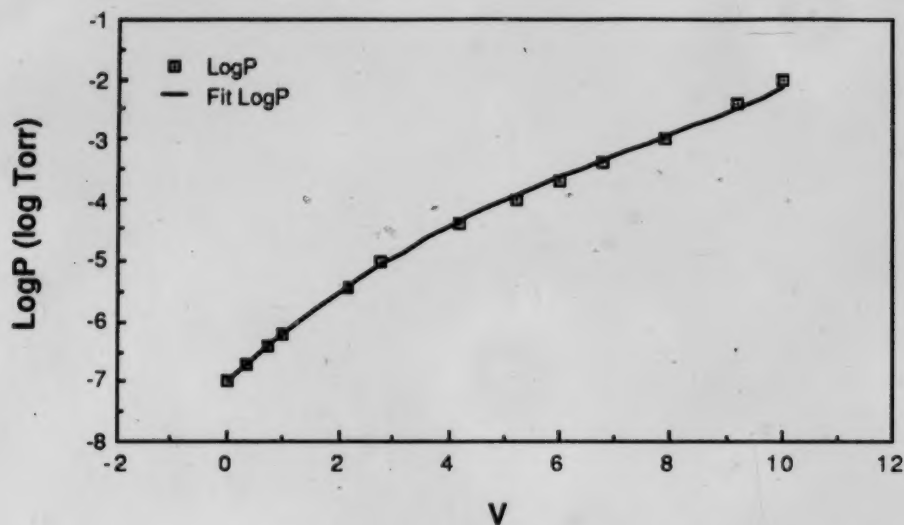


Figure V-2a: Log(pressure) vs. Gauge Voltage with polynomial fit for vacuum gauges GP1 and GP4

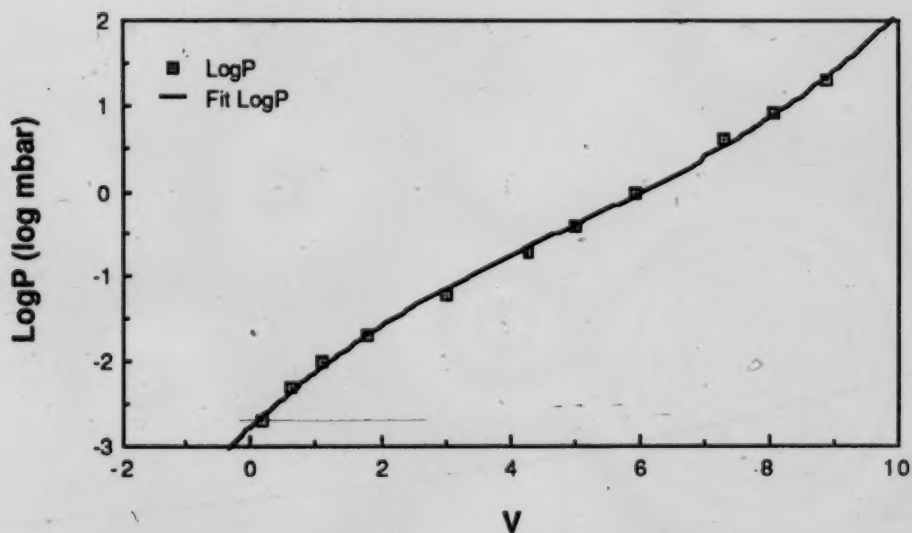


Figure V-2b: Log(pressure) vs. Gauge Voltage with polynomial fit for vacuum gauges GP2, GP3, GP5, GP6, GP7.

deleted. The second bug has not been resolved. Whenever the target system is in cooldown or warm-up, its configuration and data points do not conform with normal running. Hence a fair number of alarms are triggered if they were not turned off (e.g. by "A" in command mode). In the ensuing cascade of interrupt dumps it is next to impossible to get command "A" to be recognized to shut down the alarms, since alarms immediately call the Alphameric screen. The only suggestion at this time is to put "A" in the Alphameric mode command menu.

References

¹Private communication with Jan Soukup, May, 1986, after preliminary cooldown runs at Saclay.

²Lon Poole, Apple II User's Guide, 2nd ed (Berkley, CA: Osborne/McGraw-Hill, 1983), p. 396.

Bibliography

Primary Sources

Alifanov, O. M. et. al. "Hardware and Software Structures for Processing Thermophysics Experimental Data." Journal of Engineering Physics, 45, #5 (1983), 1223-1226.

Apple II Extended 80-Column Text Card Supplement Manual. n.p., 1985.

Apple II Reference Manual. n.p., 1981.

Busse, E. et. al. "Data Acquisition and Monitor system HOOPSY at the Heavy Ion Accelerator VICKSI." IEEE Transactions in Nuclear Science, 28, #5 (1981), 3674-3679.

Collins, John et. al. "The IUCF Inter-Computer Communication system LINKER." IEEE Transactions in Nuclear Science, 28, #5 (1981), 3684-3691.

Cameron, J. M., M. E. Schulze, and W. Turchinetz. Bates Linear Accelerator Center Updated Proposal #84-17, 1986.

Darden, S. E. "Description of Polarization and Suggestions for Additional Conventions" in Polarization Phenomena in Nuclear Reactions. Eds. H. H. Barschall and W. Haeberli. Madison, Wisconsin: University of Wisconsin Press, 1971. pp. 39-55

Darden, S. E. "Polarized Spin-One Particles." American Journal of Physics, 35 (1967), 727-738.

Dixon, W. J., et al., eds. BMDP Statistical Software 1981. Berkley, CA: University of California Press, 1981.

Garcon, M. et al. "Measurement of Vector and Tensor Analyzing Powers for 191 and 395 MeV Deuteron Scattering." Nuclear Physics, A458 (1986), 287-300.

Haeberli, W. "Polarization Experiments" in Nuclear Spectroscopy and Reactions, Part A. Ed. J. Cerny. New York: Academic Press, 1974. p. 151-191.

Ludemann, C. A. "A Microprocessor Multi-task Monitor." IEEE Transactions in Nuclear Science, 30, #5 (1983), 3858-3863.

Messiah, A. Quantum Mechanics. 2 Vol. Amsterdam: North Holland, 1962.

Ohlsen, G. "Polarization Transfer and Spin Correlation Experiments in Nuclear Physics." Report on the Progress of Physics, 35 (1972), 717-801.

Poole, Lon. Apple II User's Guide. 2nd ed. Berkley, CA: Osborne/McGraw-Hill, 1983.

SAFE BASIC Syntax and User's Manual. Revision 1.05. n.d.

Stetz, Albert W. "KIOWA - Data Histogramming and Plotting." University of Alberta, Nuclear Research Centre Internal Report #81, 1976.

Secondary Sources

Apple II DOS Manual. n.p., 1981.

Apple II Super Serial Card User's Manual. n.p., 1985.

Apple II Tutorial. n.p., 1981.

Capri, A. Z. Nonrelativistic Quantum Mechanics. Menlo Park, CA: Benjamin/Cummings, 1985.

Cooper, B. E. Statistics for Experimentalists. Oxford: Pergamon, 1969.

Frauenfelder, H. and Henley, E.M. Subatomic Physics. Englewoods Cliffs, N.J.: Prentice Hall, 1974.

IDS 460 Impact Pritter Manual. n.p., n.d.

Jackson, J. D. Classical Electrodynamics. 2nd ed. New York: Wiley, 1975.

Roy, G. "Spherical Tensor Description of Deuteron Polarization." University of Alberta, Nuclear Research Centre Internal Report #101, 1984.

Segre, Emilio. Nuclei and Particles. 2nd ed. Menlo Park, CA: Benjamin/Cummings, 1982.

Taylor, John R. An Introduction to Error Analysis. Mill Valley, CA: University Science Books, 1982.

TRIUMF Kinematics Handbook. n.d.

Appendix A: Polarization and Cross Section

There are two conventions currently in use for describing polarization states: Cartesian and Madison (spherical tensor). The Cartesian form is more convenient for describing vector polarizations and for deriving higher order polarization states. The Madison convention is convenient for higher order polarizations because its close analogy to spherical harmonics facilitates transformations in much the same way as rotating a frame of reference. Though the derivations of spin-1 polarization states are well documented (Ohlsen, Haeberli, and Darden)^{1,2,3}, a quick review is included here.

In scattering theory, the differential cross section, with respect to outgoing momentum k and scattering angle θ may be expressed in terms of a scattering amplitude, $f(k, \theta)$,⁴

$$\frac{d\sigma}{d\Omega} = \frac{|f(k, \theta)|^2}{4\pi} \quad (\text{A-1})$$

$$\Psi_f = \Psi_{\text{plane wave}} + f(k, \theta) \Psi_{\text{spherical}} \quad (\text{A-2})$$

which is a solution of the Schrodinger equation including the interaction. It follows that $f(k, \theta)$ can be expressed in Dirac notation as

$$f(k, \theta) = \langle \Psi_f | V | \Psi_i \rangle \quad (\text{A-3})$$

where V is the potential in the scattering. This quantity is also equivalent to an element T_{fi} of a matrix M called the scattering matrix. Furthermore, it is customary to include a density of states, ρ , in equation A-1, so that it looks like

$$\frac{d\sigma}{d\Omega} = \frac{|\langle \Psi_f | V | \Psi_i \rangle|^2}{4\pi} \rho \quad (\text{A-4})$$

Equivalently,

$$|\Psi_f\rangle = M |\Psi_i\rangle \quad (\text{A-5})$$

and defining initial and final densities of state (density matrices)

$$\rho_i = |\Psi_i\rangle \langle \Psi_i| \quad (\text{A-6})$$

$$\rho_f = |\Psi_f\rangle \langle \Psi_f| = M |\Psi_i\rangle \langle \Psi_i| M^* = M \rho_i M^* \quad (\text{A-7})$$

Then $\text{Tr } \rho_f$ is also the differential cross section. Note that these products are tensor products. That is, instead of the normal row-column scalar product, they are column-row products resulting in matrices. It is in the expression of the density matrices that the vector and tensor polarization appear, as coefficients of spin operators in the expansion of the density matrix.

Vector polarization is simple and easy to work with, particularly since there are at most three observables to deal with: p_x, p_y, p_z . In the case of spin-1 physics, there are nine additional cartesian observables corresponding to the product of spin operators, $S_i S_j$. These are the tensor states. But in expanding

$$S_i S_j = (1/2)(S_i S_j + S_j S_i) + (1/2)(S_i S_j - S_j S_i) \quad (\text{A-8})$$

the antisymmetric part is the well known commutation for iS_k . To separate out the vector term and to ensure that the spin matrix has a zero trace, the tensor polarization is defined as ⁵

$$p_{ij} = (3/2) \langle S_i S_j + S_j S_i \rangle - 2\delta_{ij} \quad (\text{A-9})$$

The number of observables is now reduced to six observables, and all six appear in the cross section. But this system is still overspecified. It was then realized that if the Cartesian components of spin were treated as Cartesian coordinates, then the six Cartesian tensors could be collected in three Spherical tensors in the form of spherical harmonics, Y_{ij} . That is, if $S_x = \sin\theta \cos\phi$, $S_y = \sin\theta \sin\phi$, and $S_z = \cos\theta$, and knowing that $Y_{20} \propto 3\cos^2\theta - 1$, $Y_{2\pm 1} \propto \cos\theta \sin\theta e^{\pm i\phi}$, $Y_{2\pm 2} \propto \sin^2\theta e^{\pm 2i\phi}$, constructing the spherical tensor polarizations is quite straight forward:^{6,7}

$$t_{20} = (1/\sqrt{2}) p_{zz} \quad (\text{A-10a})$$

$$t_{2\pm 1} = \pm(-1/\sqrt{3}) (p_{xz} \pm ip_{yz}) \quad (\text{A-10b})$$

$$t_{2\pm 2} = (1/2\sqrt{3}) (p_{xx} - p_{yy} \pm 2ip_{xy}) \quad (\text{A-10c})$$

And similarly for the vector polarization:

$$t_{10} = \sqrt{(3/2)} p_z \quad (\text{A-11a})$$

$$t_{11} = \pm(-\sqrt{3})/2 (p_x \pm ip_y) \quad (\text{A-11b})$$

Note that to reflect this analogy, the labelling of the Madison formalism (as it has come to be called) is the same as that for the harmonics. Taking parity into consideration, the spherical tensors satisfy

$$t_{l-m} = (-1)^{l+m} t_{lm}^* \quad (\text{A-12})$$

So the observables in a scattering experiment are t_{20} and the real parts of it_{11} , t_{21} , and t_{22} .

It should be noted that this use of spherical harmonics is applicable only to the conversion of Cartesian and Madison formalisms, and that it is only an analogy. Subsequent derivations of angular dependences are due to rotation properties of these harmonics, and should not be confused with a reapplication of another spherical harmonics analogy, regardless of the similarity of form.

Defining ζ_{lm} as the spherical analog of the cartesian spin operators,

$$\rho_i = \sum_{l,m} t_{lm} \zeta_{lm} \quad (\text{A-13})$$

Note that parity consideration

$$\zeta_{l-m} = (-1)^{l+m} \zeta_{lm}^\dagger \quad (\text{A-14})$$

results in 4 usable operators corresponding to the remaining polarizations: ζ_{20} and real parts of $i\zeta_{11}$, ζ_{21} , ζ_{22} .

Using equation A-13 in equation A-7, the differential cross section becomes

$$\begin{aligned} \sigma &= \sigma_0 \left(1 + \sum_{l,m} t_{lm} T_{lm} \right) \\ &= \sigma_0 \left[1 + 2\text{Re}(it_{11}) \text{Re}(iT_{11}) + t_{20} T_{20} \right. \\ &\quad \left. + 2\text{Re}(t_{21}) \text{Re}(T_{21}) + 2\text{Re}(t_{22}) \text{Re}(T_{22}) \right] \end{aligned} \quad (\text{A-15})$$

where $\sigma_0 = \text{Tr}(MM^\dagger)$

$$\sigma_0 T_{lm} = \text{Tr}(M \zeta_{lm} M^\dagger)$$

and the parity relations equation A-12 and equation A-14 have been used.

References

- ¹G. Ohlsen, "Polarization Transfer and Spin Correlation Experiments in Nuclear Physics," Report on the Progress of Physics, **35** (1972), 717-801.
- ²W. Haerberli, "Polarization Experiments" in Nuclear Spectroscopy and Reactions, Part A, ed. J. Cerny (New York: Academic Press, 1974), pp. 151-191.
- ³S. E. Darden, "Description of Polarization and Suggestions for Additional Conventions" in Polarization Phenomena in Nuclear Reactions, Eds. H. H. Barschall and W. Haerberli (Madison, Wisconsin: University of Wisconsin Press, 1971), pp. 39-55.
- ⁴A. Messiah, Quantum Mechanics, trans. J. Potter (Amsterdam: North-Holland, 1962), II, 801-870.
- ⁵Haerberli, p. 156.
- ⁶S. E. Darden, "Polarized Spin-One Particles," American Journal of Physics, **35** (1967), 729.
- ⁷Darden, in Polarization Phenomena, p. 41.

Appendix B: Wire-E Counter Correspondence

It was necessary to match wire chamber events against corresponding E counter events. There were a small but noticeable number of events from backscattered protons and other events recorded simultaneously with the desired event that were not caught by previous cuts. To do this matching it was necessary to know which wires could be hit and result in a valid E counter event. The obvious procedure was to do run some replays that ignored all events except the ones that fired a chosen detector. This was done for each of the four E counters. It was anticipated that twelve wire ends would display a high number of counts. This was not clearly the case in the vector polarization runs. In some cases, there were only ten. This was not serious except that if some of the replays actually reverted back to 12 wire ends some useful data would be lost. It was then realized that the number of "12" was divined from analysis of correspondences with E counters from the middle sectors of the arrays. This was quite a different situation from the present analysis which looked at the end sectors. The question was which wire ends should be added to the shorthanded sets. An attempt to ascertain this information by inspection of nearest wire end data proved unreliable. Then the geometry of the situation was investigated, thus illuminating the problem: the beam for the vector polarization runs was made narrower than the subsequent Tensor runs. In fact, according to the wire information, it was estimated that the beam spot was approximately 5cm in diameter. Front end wire chamber data verified this.

The investigation was simple, and two idealized approaches were used: a straight forward diagrammatical one, and geometric calculations. Since this was not a critical problem, uncertainties were ignored. It was assumed that the target was centered in the detector array, that the E counters were about 20.5 cm from the center of the array (records are not clear on this, nor is it easy to assess this due to an uncertain amount of material used to optically isolate the ΔE and E scintillators). Looking at E counter E15

(figure B-1), call SC the line from the sector joint between E15 and E16 and passing through the center C, and AC, the line from the hexagonal joint to C; define angle β as the angle subtended from SC to line SE (the line from S to the nearest tangent point on the beam spot with radius r), and α as the angle subtended from AC to line AD (the line from A to the nearest tangent point on the beam spot). Since the E counter array is hexagonal in cross section,

$$\sin \alpha = (\sqrt{3}) r / 41 \quad (B-1)$$

$$\sin \beta = (r / 20.5) \cos[\arctan(1/(3\sqrt{3}))] \quad (B-2)$$

What is of real interest are the angles, α' and β' , subtended by lines from C to the intersection point of the wire radii (17.62 cm) and lines AD and SE. These can be derived by the sine rule:

$$\frac{\sin \alpha}{17.62} = \frac{\sqrt{3} \sin(\alpha + \alpha')}{41} \quad (B-3)$$

$$\frac{\sin \beta}{17.62} = \frac{\cos[\arctan(1/(3\sqrt{3}))] \sin(\beta + \beta')}{20.5} \quad (B-4)$$

For a beamspot diameter of 5 cm (see figure B-2a), the additional apparent angles that the wire chamber must cover are $\alpha'=2.1^\circ$ and $\beta'=1.3^\circ$. Now the end detectors such as E15 have an angular width of

$$\pi/6 - \arctan(1/(3\sqrt{3})) = 19.1^\circ$$

This corresponds to 7 wire ends, #24 - #30, with overlaps onto E14 and E16, due to the size of the cell which the wire occupies. Note that wire end #24 wraps around to #21 which will receive counts due to #24 despite its not being in the "hit zone". At the hexagonal joint, if one assumes that wire end #30 sits right at the joint, the overlap is 1.5° . At the sector joint, the overlap would then be 0.4° . With these in mind, α' represents two additional wire end firing (#31 and #34 - the two ends of one wire of which only #31 gets a direct hit) and β' represents another two (#23 and #20) for a total of 12 wire ends firing (remembering wire ends are paired). For E14, the situation is slightly different because wire end #30 wraps around to #27, but the net result is the same

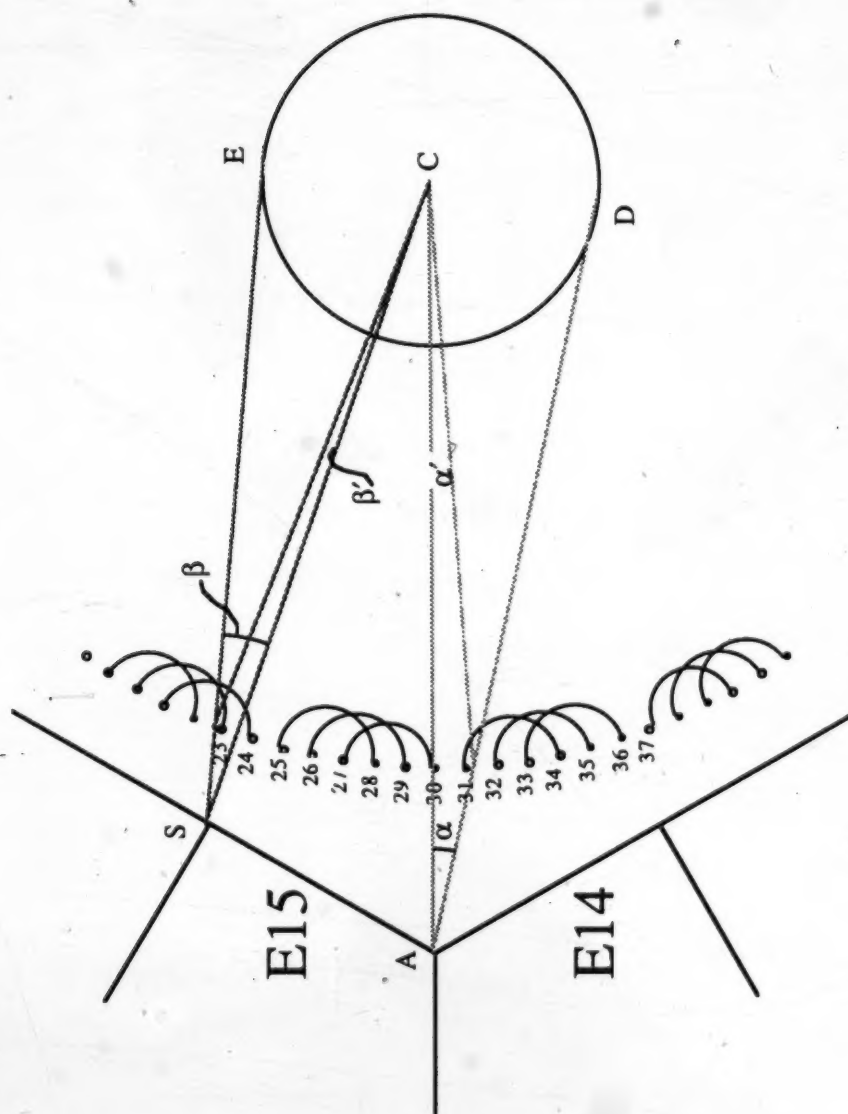


Figure B-1: Geometry for mathematical calculation of wire correspondence

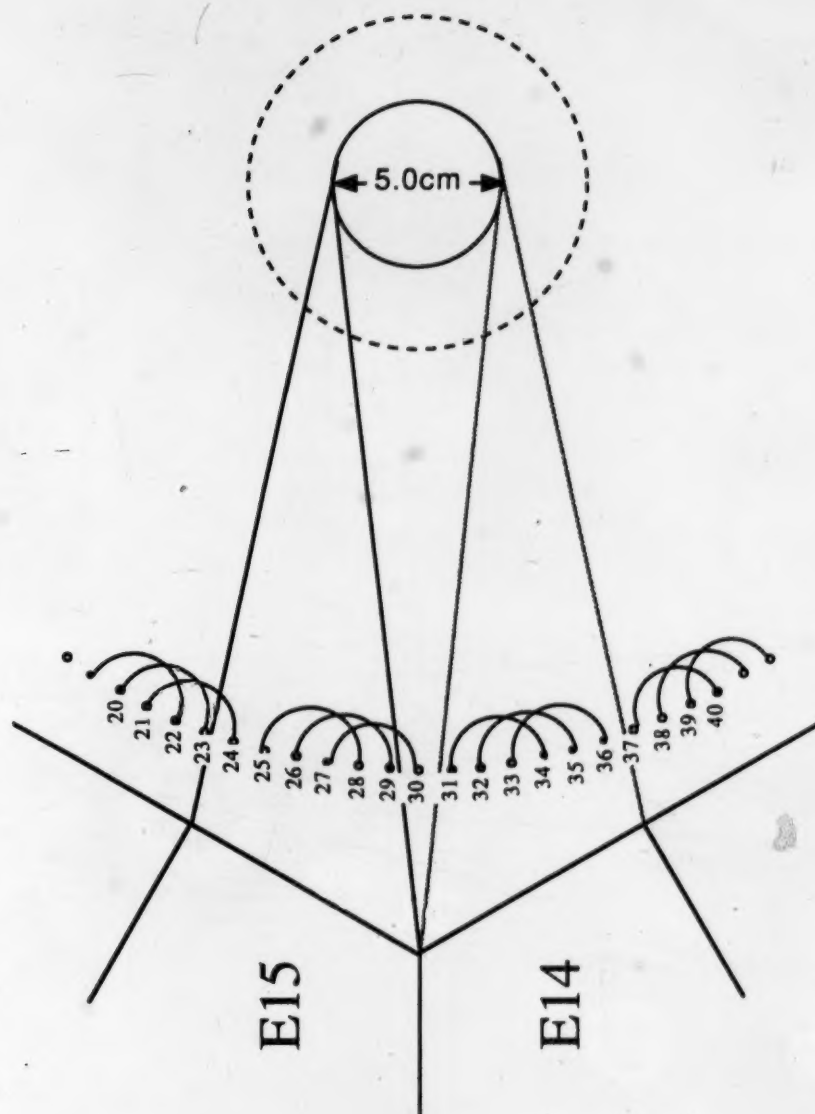


Figure B-2a: Wire Correspondences on Right Side
(viewed from downstream) with 2.5 cm radius Beamspot.

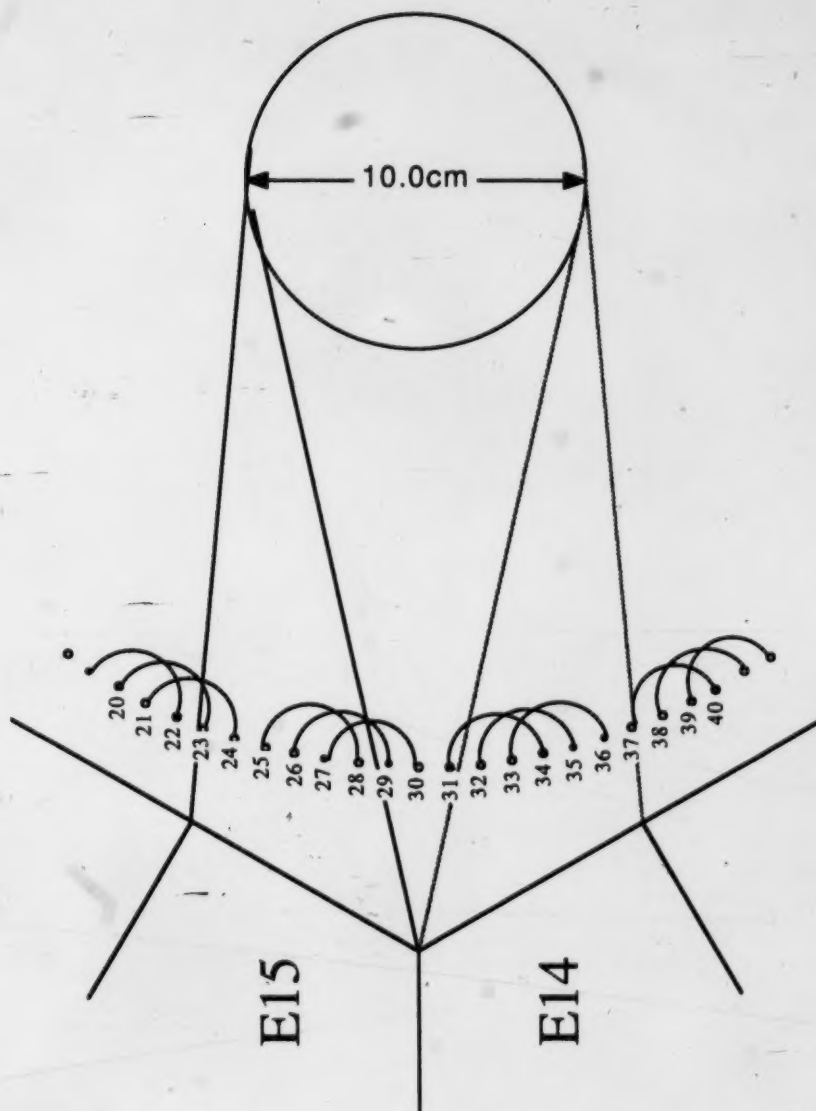


Figure B-2b: Wire Correspondences on Right Side
(viewed from downstream) with 5 cm radius Beamspot.

because there is no equivalent of #24 at the sector joint. Wire ends #29 - #36, #26, #27, #37, #40, for a total of 12 wire ends. This was not the case in the replay.

If the wire end #30 was offset from the joint by 0.5° towards E14, this would eliminate wire end #31 (and #34) at the hexagonal joint with wire end #23 at the sector joint would still be able to cover the additional angle. This totals 10 wire ends for E15, and 12 for E14, which was what was found in the wire correspondence replays.

For normal runs, where the beam spot is expected to be as near to the size of the target cell as possible (see figure B-2b), $\alpha' = 4.3^\circ$ and $\beta' = 2.6^\circ$. This represents two wire ends at the hexagonal joint and two wire ends at the sector joint for a total of 12 wire ends firing, but no offset can reduce this. The important point here is that when faced with having to decide which wire to add to the set of wires corresponding to a certain E counter, it should be the one adjacent to the last wire at the hexagonal joint (e.g. in this case, wire ends #31 and #34). Secondly, it will always be the wires at the hexagonal joint that reveals this angular variation due to the beam spot variations.

It was also of some interest as to see how the middle sectors looked in similar circumstances. Geometrically, the calculation is identical to the β' calculations (due to the symmetry about line SC). So for a diameter of 5 cm, the additional angles on either side of the E counter is 2.1° . The angular width of this detector is

$$2 \arctan(1/(3\sqrt{3})) = 21.8^\circ$$

Assuming the wire array is offset by 0.5° , this can be covered by 8 wire ends with an overlap of 2° . With the additional angles, there would be 10 wire ends directly hit by events. This is all irrelevant when one considers the pairing of wire ends (see Figure B-3). One group of 6 paired ends starts at the middle of the sector fully covering half of the detector. Another group of 6 covers the other half. The two groups always provide more than full coverage, so 12 wire ends always fire.

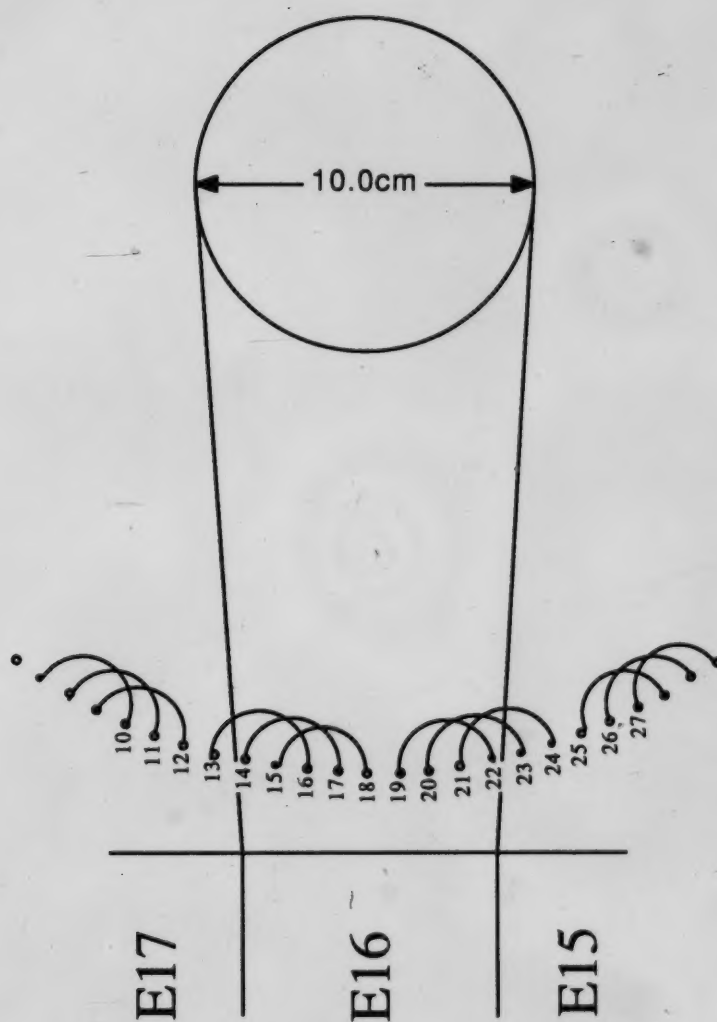


Figure B-3: Wire Correspondences on a center sector.
Note that 6 wires (12 wire ends) are necessarily involved.

Appendix C: The Least Squares Algorithm

To correlate target data from different runs, data from each run and from three devices were fit to polynomial and nonlinear functions. In neither case were the fits considered good, but they were adequate for their purpose. The biggest problem lay in the degrees of freedom available; the level sensor data consisted of 6-8 usable points. So the number of parameters had to be limited to a maximum of 4.

The basic idea is to minimize the chi-squared of the test function to the data. The chi-squared is

$$\chi^2 = \sum_i^n \frac{(y_i - f(x_i))^2}{\sigma_i^2} \quad (C-1)$$

where x_i, y_i are the i th data point

$1/\sigma_i^2$ is the weight of the i th data point

$a_1, \dots, a_n$ are the parameters of the fit

$f(x_i) = f(x_i, a_1, \dots, a_n)$ is the function to be fitted to the data

Define the residue of the j th parameter as r_j :

$$r_j = \sum_i^n \frac{(y_i - f(x_i))}{\sigma_i^2} \frac{\partial f(x_i)}{\partial a_j} \quad (C-2)$$

To minimize χ^2 , one must reduce r_j to zero. For a function linear with respect to its parameters, this is easily done by setting r_j to zero and solving the system. If a computer is available, a matrix approach to the solution is ideal. For example, a polynomial fit was used:

$$f(x_i, a_1, \dots, a_n) = a_0 + a_1 x_i + a_2 x_i^2 + \dots + a_n x_i^n \quad (C-3)$$

The derivative component of the residue is simply one term:

$$\frac{\partial f(x_i)}{\partial a_j} = x_i^j \quad (C-4)$$

Then the system to solve can be expressed as a matrix whose elements consist of the sums and products of powers of x_i and y_i :

$$\begin{aligned}
 0 &= \sum_i \frac{(y_i - a_0 - a_1 x_i - \dots - a_n x_i^n)}{\sigma_i^2} x_i^j \\
 &= \sum_i \frac{(y_i x_i^j - a_0 x_i^j - a_1 x_i^{j+1} - \dots - a_n x_i^{j+n})}{\sigma_i^2}
 \end{aligned} \quad (C-5)$$

$$0 = \begin{bmatrix} \sum 1/\sigma_i^2 & \dots & \sum x_i^n/\sigma_i^2 & \sum y_i/\sigma_i^2 \\ \sum x_i/\sigma_i^2 & \dots & \sum x_i^{n+1}/\sigma_i^2 & \sum y_i x_i/\sigma_i^2 \\ \vdots & & \vdots & \vdots \\ \sum x_i^n/\sigma_i^2 & \dots & \sum x_i^{2n}/\sigma_i^2 & \sum y_i x_i^n/\sigma_i^2 \end{bmatrix} \quad (C-6)$$

By row reducing this matrix, the parameters a_j can be read directly from the last column.

For functions not linear in its parameters (e.g. exponential), there is no exact systematic way of obtaining exact values for its parameters. What is done instead is to increment the parameters so that the residue zeros in on a zero value. How this incrementation is performed is a problem. Since the residue is an indication of how close the chi-squared is to minimum, the increment should depend on it. By differentiation one finds that

$$\Delta a_j = \frac{r_j}{\left[\frac{\partial r_j}{\partial a_j} \right]} \quad (C-7)$$

where

$$\frac{\partial r_j}{\partial a_j} = \sum_i \frac{1}{\sigma_i^2} \left[(y_i - f(x_i)) \frac{\partial^2 f(x_i)}{\partial a_j^2} - \left[\frac{\partial f(x_i)}{\partial a_j} \right]^2 \right] \quad (C-8)$$

But due to the instability of convergence, a convergence speed factor c must be introduced. The optimum value of this coefficient is 2 (hence the term "increment halving", it is suspected), but for more gradual convergence larger values can be used.

This is particularly helpful when fair initial estimates of parameters are not available because it prevents overflows and other problems related to large incrementations. So the final increment is

$$\Delta a_j = \frac{r_j}{c \left[\frac{\partial r_j}{\partial a_j} \right]} \quad (C-9)$$

Appendix D: Monitor Instructions

How to Use the SAFE 8000/Apple IIe Target Monitor (V1.0 18 May, 1987)

1. Powering up: Turn on the Apple, then the SAFE 8000. The printer must always be turned on after the Apple because it has, on occasion, rendered the Apple DOS inoperable. Make sure Caps Lock is on, on the Apple.

2. Starting Up the Monitor

- A. Insert the program disk into the disk drive.
- B. If the SAFE 8000 does not contain its monitor program, see 4.E (Trouble).
- C. Enter: **RUN H2 PC BOOT**

It will take a couple of minutes to run through the start-up routines.

- D. "Parameters menu"

"1) LIST PARAMETERS?"

Causes a dump, to screen, of parameter settings. <CTRL-S> pauses the dump, <CTRL-Q> continues. The format is as follows:

Digital: name, priority, timer, set point

Analog: name, priority, timer, low set point, high set point

If priority is ≤ 23 , the alarm is enabled at that priority. Highest priority is 23. If it is ≤ 64 but > 23 then it is disabled subject to user changes (see 2.D.2 below). Subtract 32 from its displayed priority value to find the alarm priority. If displayed priority is ≥ 64 then its alarm is permanently disabled.

Timer is the time delay (in 1/10ths of seconds) before going to the second alarm due to a prolonged and unrectified alarm.

Set points for digital is the value (0/1 = off/on) that causes an alarm, and for analog, are the values (converted from voltages to 12-bit,

where full scale = 4095) for the lowest and highest acceptable voltages from analog devices.

"2) CHANGE PARAMETERS?"

Brings up a menu to make changes to the above mentioned parameters. The format of instructions is 'menu #, new value'. In the case of "NO CHANGE" and "GO TO MAIN PROGRAM", note that the ",0" for a new value is necessary after the menu #.

"3) GO TO MAIN PROGRAM"

At this stage, this actually goes to "TIME OK?" below.

There will be a delay while information is transferred to Apple.

E. "TIME OK?"

"N" - It will ask for the correct time and also a date.

"Y" - Goes directly to 2.F. below.

If you only want to change date, answer "N" to time, then hit <RETURN> when it asks for the time. This will not affect the clock.

F. You will finally be prompted to insert a data disk. This must have been Apple DOS formatted, but no special files are needed.

3. System Running

A. "Numeric" mode (display) is the first screen up. Also, when an alarm is triggered, the display goes automatically to "Numeric".

B. Every mode has a menu at the bottom of the screen. When the indicator at the bottom, right-hand corner of the screen is lit, pressing any of the indicated keys causes the Apple to go to the indicated mode, or perform the indicated function. There usually a delay (about 4 s), but if the indicator turns off and there is still no response, the program has been corrupted. At this time, hitting <CTRL-RESET> seems to be the only way out, but give it a chance (if you have the time).

C. Brief description of Screen Modes

- I) **N** - Numeric: Displays data in text form.
- II) **G** - Graph: Displays analog data in x - y graphs, 4x4, with 40 x 40 pixel resolution.
- III) **H** - Horizontal Bar Graphs: analog data with resolution of 30 increments.
- IV) **R** - Raw Data: Outputs data as voltage and digital on/off (1/0) data as measured at the input cards.
- V) **C** - Command: Besides accessing other modes:
 - M** - brings up the parameter modify routine (see 2.D)
 - S** - Sets time and/or date (see 2.E)
 - P** - Performs one data dump to printer
 - T** - Toggles periodic printer dump enable (see also "I")
 - I** - Sets printer dump interval
 - F** - Lists files on disk
 - D** - Toggles periodic disk dump enable
 - A** - Toggles all available alarms in one stroke
 - O** - Allows control of selected switches (e.g. valves)
 - B** - Toggles the audible alarm (i.e. the "beeping")
- D. **X** - shuts down the apple program but leave the SAFE 8000 running in manual mode. This facilitates program changes to the Apple program. To restart the Apple, enter: **RUN**
- E. **Q** - Shuts down the whole monitor system, but the control panel should still be able to go to **AUTO**.

4. Trouble

Definition: **CRASH**: A "?" (question mark) followed by a flashing box in the upper left corner of the screen; screen has not updated for at least 2 minutes (use a timer); the classic screen garbled or continuous output of garbage data.

- A. Hit <CTRL-RESET>. This stops the Apple program dead.
- B. Enter **RUN**. Since this resets the Apple program driver, it means that you loose all data since the last disk dump. The system should go to 2.D. above.
- C. If it doesn't, hit <CTRL-RESET>, then go directly to SAFE (see 4.E.) and hit "Q" then "S". If it doesn't respond with a data dump followed by "OK", then hit <CTRL-C>. In that case, it should respond with "BREAK AT ####" and/or "OK". Return to Apple, and **RUN**.
- D. If things still don't work, restart the whole system by hitting <EMPTY APPLE-CTRL-RESET>, which boots Apple DOS, and then go to 2.C. (or 4.E., if necessary)
- E. The SAFE 8000 has a battery backed memory, so its monitor program should always be in memory, even when powered down. But if it has been found that the SAFE 8000 does not contain its monitor program, enter

RUN SAFE LOADER

It will ask which file is to be dumped to SAFE. Respond with

H2 SAFE.TEXT

It will take a few minutes. Beware of line noise. If the SAFE program crashes after loading, check the program listing and make corrections, or try loading again.

To communicate with SAFE directly (i.e. Apple as dumb terminal), enter

IN#2 (RETURN)

<CTRL-A><T><RETURN>

The program in SAFE can be edited in this mode (see SAFE BASIC manual).

To leave (or reset the Super Serial Card), enter

<CTRL-A><R><RETURN>

(Note: the last two items here assumes the Apple is using an Apple Super Serial Card)

To leave but keep the card active (in case it has been reconfigured by software, as opposed by hardware switches), enter

<CTRL-A><Q><RETURN>

- F. For further details on the SAFE 8000, refer to the SAFE BASIC Syntax and User's Manual. For further details on Apple terminal settings, refer to the Apple II Super Serial Card User's Manual.

5. "On the disk..."

- A. Currently, there is no external, permanent memory for the SAFE 8000.

Therefore, programs written in the SAFE 8000 must be sent to the Apple, which copies to disk. Unfortunately, a program in Apple is necessary to transfer to disk, and in BASIC this is slow. To save to disk, enter

RUN SAFE LISTMAKER

It is necessary to use port#2 at 300 baud on the SAFE, though changing the program to use port#1 is possible (requires that port#1 be set to 300 baud first - in terminal mode enter **OPEN COM#1,0** to set to 300 baud, and **OPEN COM#1,2** to set to 1200 baud). Note that both machines must be set at 300 baud (the Apple is changed by **<CTRL-A><6B><RETURN>** to set to 300 baud, and **<CTRL-A><8B><RETURN>** to set to 1200 baud. Note also that the disk copy is a text file.

- B. The use of "SAFE LOADER" has been outlined in 4.E.

- C. There is currently no way of directly printing out SAFE 8000 programs from memory via Apple. Hence a program is needed. Enter

RUN SAFE PRINTTEXT

to print out the current text file on disk, or

RUN SAFE PRINTOUT

to print out the program in SAFE memory. For this facility, it is also necessary to set the speed to 300 baud first (see 5.A).

Appendix E: Program Listings

E.1: H2 SAFE (SAFE 8000 monitor program listing)

```
10 CLR:MC$="E" ;SAFE AUTOSTART
20 GOTO 4242
30 REM REVISED 10/07/86
40 OPEN COM#2,0:STOP
50 REM START UP ;GO TO NORMAL START UP
60 GOTO 4240
70 REM START UP CONTD
80 SETTASK 490,1:SETTASK 620,2:SETTASK 720,3
90 GAIN 0,28,0
100 INTERVAL 123,10:I1=1 ;SET TIMER INTERVALS
110 INTERVAL 126,150:I2=60
120 SETTIMER 123,I1
130 TIMEOUT 0,1,126,2
140 ON COM#1 GOSUB 1230 ;SET COMMUNICATION ROUTINE LOC
144 ON ERR GOSUB 2670
150 TURNON 2,7
160 ON IRQ-23 GOSUB 2650 ;SET INTERRUPT ROUTINE LOCS
170 ON IRQ 22 GOSUB 1560
180 ON IRQ 21 GOSUB 1570
190 ON IRQ 20 GOSUB 1580
200 ON IRQ 19 GOSUB 1590
210 ON IRQ 18 GOSUB 1600
220 ON IRQ 17 GOSUB 1610
230 ON IRQ 16 GOSUB 1620
240 ON IRQ 15 GOSUB 1630
250 ON IRQ 14 GOSUB 1640
260 ON IRQ 13 GOSUB 1650
270 ON IRQ 12 GOSUB 1660
280 ON IRQ 11 GOSUB 1670
290 ON IRQ 10 GOSUB 1680
300 ON IRQ 9 GOSUB 1690
310 ON IRQ 8 GOSUB 1700
320 ON IRQ 7 GOSUB 1710
330 ON IRQ 6 GOSUB 1720
340 ON IRQ 5 GOSUB 1730
350 ON IRQ 4 GOSUB 1740
360 ON IRQ 3 GOSUB 1750
370 ON IRQ 2 GOSUB 1760
380 ON IRQ 1 GOSUB 2610
390 ON IRQ 0 GOSUB 1810
400 TURNON 23,4
410 REM MAIN ROUTINE
420 IF MC$="M"THEN SETTIMER 125,3:GOTO 2770 ;SCANNING LEVEL COMMANDS
430 IF MC$="T"THEN SETTIMER 125,3:GOTO 3640 ;MC$=COMMAND CODE FROM COMM
440 IF MC$="O"THEN SETTIMER 125,3:GOTO 4054
442 IF MC$="E"THEN 4222
450 IF MC$<>"Q"THEN 470
460 UNTIL IT=0:REPEAT:END
470 TASK:GOTO 420
480 REM TIMER TASK ;TIMERS SCAN TASK.
490 T=0
500 IF DA(T)=0 THEN 530
510 IF TIMER(T)=0 THEN LD(T)=1 ;LD=DIG LATE RESP ALARM FLAG
520 GOTO 540
```



```

530 SETTIMER T,0:LD(T)=0
540 IF T>16 THEN 590
550 IF AA(T)=0 THEN 580
560 IF TIMER(T+64)<15 THEN LA(T)=1
570 GOTO 590
580 SETTIMER T+64,0:LA(T)=0
590 TASK
600 T=T+1:IF T>64 THEN 490:ELSE GOTO 500
610 REM AIN TASK
620 A=0
630 IF AP(A)>23 THEN 690
640 AD(A)=AIN(A)
650 IF AD(A)>AL(A) AND AD(A)<AU(A) THEN AA(A)=0:GOTO 690
660 GR=AP(A):IF AA(A) THEN 680
670 AA(A)=1:GC(GR)=GC(GR)+1
680 IG(GR,GC(GR))=A:TURNON AC(A),AB(A)
690 TASK
700 A=A+1:IF A<16 THEN 630:ELSE GOTO 620
710 REM DIN TASK
720 D=0
730 IF DP(D)>23 THEN 800
740 DS=INT(D/8):DG=BT(D-DS*8)
750 DD(D)=(DIN(DS+4) AND DG)/DG
760 IF DD(D) EOR DI(D) THEN DA(D)=0:GOTO 800
770 GR=DP(D):IF DA(D) THEN 790
780 DA(D)=1:GC(GR)=GC(GR)+1
790 IG(GR,GC(GR))=D:TURNON DC(D),DB(D)
800 TASK:D=D+1
810 IF D<18 THEN 730
820 D=24:GOSUB 990:TASK
830 D=25:GOSUB 990:TASK
840 D=33:GOSUB 1070:TASK
850 D=35:GOSUB 1070:TASK
860 D=37:GOSUB 1070:TASK
870 D=39:GOSUB 1070:TASK
880 D=42:GOSUB 1070:TASK
890 D=45:GOSUB 1070:TASK
900 D=48:GOSUB 1070:TASK
910 D=51:GOSUB 1070:TASK
920 D=54:GOSUB 1070:TASK
930 D=58:GOSUB 1070:TASK
940 D=60:GOSUB 990:TASK
950 D=61:GOSUB 990:TASK
960 IF DIN(22) AND DIN(23) OR (AMS="M") THEN 980
970 IG(0,1)=64:DA(64)=1:TURNON 0,0:TASK
980 GOTO 720
990 IF DP(D)>23 THEN 1060
1000 DS=INT(D/8):DG=BT(D-DS*8)
1010 DD(D)=(DIN(DS+4) AND DG)/DG
1020 IF DD(D) EOR DI(D) THEN DA(D)=0:GOTO 1060
1030 GR=DP(D):IF DA(D) THEN 1050
1040 DA(D)=1:GC(GR)=GC(GR)+1
1050 IG(GR,GC(GR))=D:TURNON DC(D),DB(D)
1060 RETURN
1070 DS=INT(D/8):DG=BT(D-DS*8)
1080 DD(D)=(DIN(DS+4) AND DG)/DG
1090 DS=INT((D+1)/8):DG=BT(D+1-DS*8)
1100 DD(D+1)=(DIN(DS+4) AND DG)/DG
1110 IF DP(D+1)>23 THEN 1160

```

;LA=ANAL LATE RESP ALARM FLAG

;ANALOG SCAN TASK

;AP=ANALOG PRIORITY LEVEL
;AD=ANAL DATA
;AL&AU=LO&HI ANAL SET PTS
;AA=ANAL ALARM FLAG
;GC=INTERRUPT (LEVEL) COUNTER
;IG=INTERRUPT VARIABLE TAG

;DIGITAL SCAN TASK

;DP=DIGITAL PRIO LEVEL
;DS=DIG SLOT#,DG=DIG BIT#
;DD=DIG DATA
;DI=DIG SET PT
;DA=DIG ALARM FLAG

;VALVE MONITORING ROUTINE

```

1120 IF DD(D)EOR DD(D+1)THEN DA(D+1)=0:GOTO 1160
1130 GR=DP(D+1):IF DA(D+1)THEN 1150
1140 DA(D+1)=1:GC(GR)=GC(GR)+1
1150 IG(GR,GC(GR))=D+1:TURNON DC(D+1),DB(D+1)
1160 IF DP(D)>23 THEN 1210
1170 IF DD(D)EOR DI(D)THEN DA(D)=0:GOTO 1210
1180 GR=DP(D):IF DA(D)THEN 1200
1190 DA(D)=1:GC(GR)=GC(GR)+1
1200 IG(GR,GC(GR))=D:TURNON DC(D),DB(D)
1210 RETURN
1220 REM COM INTERRUPT
1230 SETTIMER 124,10
1240 GET HCS ;INPUT READY OR COMMAND CODE
1250 IF TIMER(124)>0 AND (HCS<"A"OR HCS>"Z")THEN 1240 ; FROM APPLE II
1260 IF HCS="Q"THEN 2670
1270 IF HCS="X"THEN 1350
1280 IF HCS="R"THEN 1360
1290 IF HCS="I"THEN 3970
1300 IF HCS="A"THEN 3820
1310 IF HCS="M"OR HCS="T"THEN 1370
1320 IF HCS="O"THEN 1370
1330 IF HCS="S"THEN 1380
1340 GET GR$:GET GR$:PRINT" N":RETURN
1350 IT=0:AM=1:GOTO 3820
1360 HCS="":GET GR$:PRINT" X"
1362 IT$(IT)=" PP"+CR$+TI$+CR$+DA$+CR$+"$":IT=IT+1:RETURN
1370 MCS=HCS:IRQ OFF 24:RETURN
1380 IF IT=0 THEN 1440 ;ANY INTERRUPTS?
1390 PRINT" Y":HCS="":GET GR$
1400 FOR G=0 TO 100:NEXT
1410 PRINT IT$(CI):CI=CI+1 ;IT$=INTERRUPT MESSAGES
1420 IF CI>=IT THEN IT=0:CI=0
1430 HCS="":RETURN
1440 IF TIMER(123)<>0 THEN PRINT" N":RETURN ;DATA TRANSMISSIONS
1450 GET GR$:GET GR$
1460 PRINT" Y":HCS=" "
1470 IF C>0 THEN C=0:GOTO 1510
1480 PRINT" DD";CR$;DIN(4);CR$;DIN(5);CR$;DIN(6);CR$;DIN(7);CR$;
1490 PRINT DIN(8);CR$;DIN(9);CR$;DIN(10);CR$;DIN(11);CR$;"$ "
1500 C=C+1:SETTIMER 123,1:HCS="":RETURN
1510 PRINT" DA";CR$;:FOR CA=0 TO 15
1520 PRINT AIN(CA);CR$;
1530 NEXT CA:PRINT" $":SETTIMER 123,11
1540 HCS="":RETURN
1550 REM SPECIAL INTERRUPT ROUTINES
1560 N=22:GOTO 1770 ;DIRECT TO SPECIFIC
1570 I=IG(21,1):GOTO 1934 ;INTERRUPT ROUTINES
1580 N=20:GOTO 2290
1590 N=19:GOTO 2290
1600 N=18:GOTO 1770
1610 N=17:GOTO 1770
1620 N=16:GOTO 1770
1630 N=15:GOTO 2180
1640 N=14:GOTO 2180
1650 N=13:GOTO 2180
1660 N=12:GOTO 1770
1670 N=11:GOTO 1770
1680 N=10:GOTO 1770
1690 N=9:GOTO 1770

```

```

1700 N=8:GOTO 1770
1710 N=7:GOTO 1770
1720 N=6:GOTO 1770
1730 N=5:GOTO 1770
1740 N=4:GOTO 1770
1750 N=3:GOTO 1770
1760 N=2:GOTO 1770
1770 K=1:UNTIL K>GC(N):I=IG(N,K)
1780 IF I>15 THEN 1800
1790 IF AA(I) THEN 2450
1800 IF DA(I) THEN 1840:ELSE RETURN
1810 I=IG(0,1):IF I<>64 THEN RETURN:ELSE GOTO 2550
1820 REM INTERRUPT ROUTINES
1830 REM DIGITAL
1840 TURNOFF DC(I),DB(I)
1850 IF LD(I) THEN ICS=" L":ELSE ICS=" I"
1860 IF TIMER(I)>0 THEN 1910
1870 SETTIMER I,TL(I)
1880 ITS(IT)=ICS+"D"+CR$+DN$(I)+CR$+STR$(I)+CR$
1890 ITS(IT)=ITS(IT)+STR$(DI(I))+CR$+TIMES+CR$+DATES+CR$+"$"
1900 IT=IT+1
1910 K=K+1
1920 REPEAT:GC(N)=0:RETURN
1930 REM CRYO INTERRUPT
1934 K=1:UNTIL K>GC(21):I=IG(21,K)
1940 TURNOFF 2,5:IF I=14 THEN 2020
1950 IF LA(0) THEN ICS=" L":ELSE ICS=" I"
1960 IF TIMER(64)>0 THEN 2162
1970 IF AIN(0)>2764 THEN 2090
1980 IF AIN(1)<844 THEN 2080
1990 IF DIN(19) THEN 2070
2000 IF DIN(14) THEN 2020
2010 ITS(IT)=CR$+"SUSP H2 LEAK"+CR$+"0"+CR$+"0":GOTO 2150
2020 IF LD(14) THEN ICS=" L":ELSE ICS=" I"
2030 IF TIMER(14)>0 THEN 2162
2040 ITS(IT)=ICS+"D"+CR$+"H2 LEAK"+CR$+"14"+CR$+"1"
2050 ITS(IT)=ITS(IT)+CR$+TIS+CR$+DAS+CR$+"$"
2060 SETTIMER 14,TL(14):IT=IT+1:GOTO 2162
2070 ITS(IT)=CR$+"H2 IN VAC"+CR$+"3"+CR$+STR$(AIN(3)):GOTO 2150
2080 ITS(IT)=CR$+"TH1 LO"+CR$+"0"+CR$+STR$(AIN(0)):GOTO 2150
2090 IF AIN(0)<AU(0) THEN 2162
2100 IF AIN(1)<922 THEN 2140
2110 IF DIN(19) THEN 2130
2120 ITS(IT)=CR$+"TH1&TH2 HI"+CR$+"0"+CR$+STR$(AIN(0)):GOTO 2150
2130 ITS(IT)=CR$+"TH1&TH2 HI GP1 BAD"+CR$+"0"+CR$+STR$(AIN(0)):GOTO 2150
2140 ITS(IT)=CR$+"FILL PRES HI"+CR$+"1"+CR$+STR$(AIN(1))
2150 ITS(IT)=ICS+"A"+ITS(IT)+CR$+TIS+CR$+DAS+CR$+"$"
2160 SETTIMER 64,TL(64):IT=IT+1
2162 K=K+1
2163 REPEAT:GC(21)=0:RETURN
2170 REM VALVE HALF OPEN
2180 K=1:UNTIL K>GC(N):J=IG(N,K)
2190 TURNOFF DC(J),DB(J)
2200 IF LD(J) THEN ICS=" L":ELSE ICS=" I"
2210 IF TIMER(J)>0 THEN 2260
2220 SETTIMER J,TL(J)
2230 ITS(IT)=ICS+"D"+CR$+DN$(J)+" HALF OPEN"+CR$+STR$(J)
2240 ITS(IT)=ITS(IT)+CR$+"-1"+CR$+TIS+CR$+DAS+CR$+"$"
2250 IT=IT+1

```

; ID INTERRUPT DATA PT

; GENERAL DIG INTPTS

; ICS=INTPT CODE

; COMPILER & QUEUE INTPT MESSAGE

; CRYOGENICS INTPTS

; HALF-OPEN VALVE INTPTS

```

2260 K=K+1
2270 REPEAT:GC(N)=0:RETURN
2280 REM PRESS GAUGES ;PRESSURE GAUGE INTPTS
2290 K=1:UNTIL K>GC(N):I=IG(N,K)
2300 TURNOFF AC(I),AB(I)
2310 IF LA(I)THEN ICS=" L":ELSE ICS=" I"
2320 IF TIMER(I+64)>0 THEN 2420
2330 SETTIMER I+64,TL(I+64):J=15
2340 IF (LEFT$(DN$(J),3)<>ANS(I))OR(J>23)THEN J=J+1:GOTO 2340
2350 IF J=24 THEN 2370:ELSE DS=INT(D/8)
2352 DG=BT(D-DS*8):DD(D)=(DIN(DS+4)AND DG)/DG
2360 IF INT(DD(D))=DI(J)THEN 2390
2370 ITS(IT)=ICS+"A"+CRS+"SUSP "+ANS(I)+CRS+STR$(I)+CRS
2380 GOTO 2400
2390 ITS(IT)=ICS+"A"+CRS+ANS(I)+CRS+STR$(I)+CRS
2400 ITS(IT)=ITS(IT)+STR$(AD(I))+CRS+TIS+CRS+DAS+CRS+"$"
2410 IT=IT+1
2420 K=K+1
2430 REPEAT:GC(N)=0:RETURN
2440 REM ANALOG ;ANALOG INTPTS
2450 TURNOFF AC(I),AB(I)
2460 IF LA(I)THEN ICS=" L":ELSE ICS=" I"
2470 IF TIMER(I+64)>0 THEN 2520
2480 SETTIMER I+64,TL(I+64)
2490 ITS(IT)=ICS+"A"+CRS+ANS(I)+CRS+STR$(I)+CRS
2500 ITS(IT)=ITS(IT)+STR$(AD(I))+CRS+TIMES+CRS+DATES+CRS+"$"
2510 IT=IT+1
2520 K=K+1
2530 REPEAT:GC(N)=0:RETURN
2540 REM MANUAL TRIPPED
2550 IG(0,1)=0:TURNOFF 0,0
2560 IF TIMER(80)>0 THEN RETURN:ELSE SETTIMER 80,300
2570 ITS(IT)="ID"+CRS+"AUTO NO LONGER ON"+CRS+"64"
2580 ITS(IT)=ITS(IT)+CRS+"0"+CRS+TIS+CRS+DAS+CRS+"$"
2590 IT=IT+1:RETURN
2600 REM DUMP TIMEOUT
2610 TIMEOUT 0,1,126,12
2620 ITS(IT)=" PP"+CRS+TIMES+CRS+DATES+CRS+"$"
2630 IT=IT+1:TURNOFF 0,1:RETURN
2640 REM SHUT DOWN
2650 DOUT 22,7:DOUT 23,0
2660 FOR I=0 TO 23:IRQ OFF I:NEXT
2670 FOR I=0 TO 16:LA(I)=0:LD(I)=0:NEXT
2680 FOR I=17 TO 64:LD(I)=0:NEXT
2690 TURNOFF 23,4
2700 DOUT 0,0:DOUT 1,0:DOUT 2,128
2710 FOR T=Q TO 80:SETTIMER T,0:NEXT T
2720 SETTIMER 126,0:SETTIMER 123,0
2730 ITS(IT)=" QQ"+CRS+TIMES+CRS+DATES+CRS+"$"
2740 MCS="Q":IT=IT+1:RETURN
2750 END
2760 REM PARAM CHNG REQ ;PARAMETER CHANGE ROUTINE
2770 TASK:IF TIMER(125)>0 THEN 2770
2780 IRQ OFF 24
2790 PRINT TS:TASK:PRINT CRS
2800 PRINT:PRINT"WOULD YOU LIKE TO:"
2810 PRINT"1) LIST PARAMETER SETTINGS?"
2820 PRINT"2) CHANGE A PARAMETER?"
2830 PRINT"3) GO TO MAIN PROGRAM?"

```

```

2840 GET P$:IF P$=""THEN TASK:GOTO 2840
2850 PRINT " ";P$:PRINT
2860 PC=VAL(P$):ON PC GOTO 2880,2990,3480
2870 GOTO 2800
2880 PRINT"CTRL-S PAUSES, CTRL-Q CONT":PRINT
2890 PRINT"NAME,PRI0,TIMER,SET PT"
2900 FOR I=0 TO 63:PRINT DNS(I);DP(I);TL(I);DI(I)
2910 NEXT I:PRINT"NAME,PRI0,TIMER,LOWER&UPPER SET PT"
2920 FOR I=0 TO 15
2930 PRINT AN$(I);AP(I);TL(I+16);AL(I);AU(I):NEXT I
2940 GOTO 2800
2950 PRINT"WOULD YOU LIKE TO CHANGE ANY OF"
2960 PRINT"THE PARAMETERS? (Y/N)"
2970 GET YNS:IF YNS="N"THEN 3480
2980 IF YNS<>"Y"THEN TASK:GOTO 2970
2990 CNS="" :PBS="" :PRINT"WHICH ONE?"
3000 GET P$:IF P$<>" "THEN 3020
3010 TASK:GOTO 3000
3020 PRINT P$;:IF P$=CHR$(13)THEN 3040
3030 CNS=CNS+P$:GOTO 3000
3040 I=0
3050 IF DNS(I)=CNS THEN AT=0:GOTO 3110
3060 IF I<64 THEN I=I+1:GOTO 3050
3070 I=0
3080 IF AN$(I)=CNS THEN AT=1:GOTO 3120
3090 IF I<16 THEN I=I+1:GOTO 3080
3100 PRINT"I CAN'T FIND ";CNS;". NONETHELESS,":GOTO 2950
3110 PRINT"DIGITAL: ";DNS(I);DP(I);TL(I);DI(I):PRINT:GOTO 3130
3120 PRINT"ANALOG: ";AN$(I);AP(I);TL(I+16);AL(I);AU(I):PRINT
3130 PRINT"1) NAME (ANY - SHORT AS POSS.)"
3140 PRINT"2) ENABLE (0-OFF,1-ON)"
3150 PRINT"3) TIMER LENGTH (<256)"
3160 PRINT"4) DIGITAL INTERRUPT VALUE (0/1)"
3170 PRINT"5) UPPER SET PT (0-4096)"
3180 PRINT"6) LOWER SET PT (0-4096)"
3190 PRINT"7) NO CHANGE (7,0)"
3200 PRINT"8) GO TO MAIN PROGRAM (8,0)"
3210 PRINT"ENTER CHOICE AND VALUE (EG. 5,300)"
3220 GET P$:IF P$<>" "THEN PRINT P$;:GOTO 3240
3230 TASK:GOTO 3220
3240 IF P$=CHR$(13)THEN PV=VAL(PBS):GOTO 3270
3250 IF P$=","THEN PC=VAL(PBS):PBS="":GOTO 3220
3260 PBS=PBS+P$:GOTO 3220
3270 ON PC GOTO 3340,3360,3300,3310,3320,3330,2800,3480
3280 PRINT"I THINK YOU MADE A MISTAKE.":PBS=""
3290 PRINT"PLEASE TRY AGAIN.":PRINT:GOTO 3130
3300 TL(I+AT*16)=PV:GOTO 3410
3310 DI(I)=PV:GOTO 3410
3320 AU(I)=PV:GOTO 3410
3330 AL(I)=PV:GOTO 3410
3340 IF AT=1 THEN AN$(I)=PV$;ELSE DNS(I)=PV$
3350 GOTO 3410
3360 IF AT=1 THEN 3390
3370 DP(I)=(DP(I)OR 32)EOR(32*PV):IF PV>23 THEN 3420
3380 DC(I)=INT(DP(I)/8):DB(I)=DP(I)AND 7:GOTO 3420
3390 AP(I)=(AP(I)OR 32)EOR(32*PV):IF PV>23 THEN 3430
3400 AC(I)=INT(AP(I)/8):AB(I)=AP(I)AND 7
3410 IF AT=1 THEN 3430
3420 PRINT DNS(I);DP(I);TL(I);DI(I):GOTO 3450

```

```

;GET-TASK COMBINATION TO
; REPLACE INPUT FUNCTION

```



```

3430 PRINT AN$(I);AP(I);TL(I+16);AL(I);AU(I)
3440 GOTO 2800
3450 PRINT"ANOTHER PARAMETER? (Y/N)"
3460 GET YN$:IF YN$="Y"THEN 2800
3470 IF YN$<>"N"THEN TASK:GOTO 3460
3480 PRINT RS:SETTIMER 125,5
3490 TASK:IF TIMER(125)>0 THEN 3490
3500 PRINT"    CONT"
3510 REM SUB: DUMP ENABLES (1=ON)
3520 SETTIMER 125,5
3530 TASK:IF TIMER(125)>0 THEN 3530
3540 FOR I=0 TO 63
3550 IF DP(I)>23 THEN PRINT"    0":ELSE PRINT"    1"
3560 NEXT:PRINT"    $":FOR I=0 TO 15
3570 IF AP(I)>23 THEN PRINT"    0":ELSE PRINT"    1"
3580 NEXT:PRINT"    $"
3590 TASK:MC$="":TASK:ON COM#1 GOSUB 1230
3600 SETTIMER 125,10
3610 TASK:IF TIMER(125)>0 THEN 3610
3620 GOTO 470
3630 REM TIME CHANGE REQ
3640 TASK:IF TIMER(125)>0 THEN 3640
3650 IRQ OFF 24
3660 PRINT TS:TASK:PRINT CRS
3670 PRINT LEFT$(TIMES,2)+" ":"+MID$(TIMES,3,2)+" ":"+RIGHT$(TIMES,2)
3680 PRINT"IS THIS TIME OK? (Y/N)"
3690 GET YN$:IF YN$="Y"THEN 3780
3700 IF YN$<>"N"THEN TASK:GOTO 3690
3710 INPUT"WHAT TIME IS IT? (HHMMSS)",TIMES
3720 PRINT RIGHT$(DATES,2)+" "/" +MID$(DATES,3,2)+" "/" +LEFT$(DATES,2)
3730 PRINT"IS THIS DATE OK? (Y/N)"
3740 GET YN$:IF YN$="Y"THEN 3670
3750 IF YN$<>"N"THEN TASK:GOTO 3740
3760 INPUT"WHAT IS THE DATE? (YYMMDD)",DATES
3770 GOTO 3720
3780 PRINT RS:PRINT"    CONT"
3790 TASK:MC$="":ON COM#1 GOSUB 1230
3800 GOTO 470
3810 REM ALL ALARM TOGGLE
3820 AM=AM EOR 1
3830 FOR I=0 TO 63:DP(I)=(DP(I)OR 32)EOR(32*AM)
3840 IF DP(I)>23 THEN 3860
3850 DC(I)=INT(DP(I)/8):DB(I)=DP(I)AND 7
3860 DA(I)=0:NEXT:FOR I=0 TO 15:AA(I)=0
3870 AP(I)=(AP(I)OR 32)EOR(32*AM)
3880 IF AP(I)>23 THEN 3900
3890 AC(I)=INT(AP(I)/8):AB(I)=AP(I)AND 7
3900 NEXT
3910 FOR I=0 TO 23:GC(I)=0:NEXT
3920 IF AM=1 THEN 3940:ELSE PRINT"    0"
3930 DOUT 0,0:DOUT 1,0:DOUT 2,128:RETURN
3940 PRINT"    1"
3950 RETURN
3960 REM CHANGE PRINT DUMP INTVL
3970 I2$=""
3980 GET GR$:IF GR$<" "THEN 3980
3990 IF GR$=" $"THEN 4010
4000 I2$=I2$+GR$:GOTO 3980
4010 I2=VAL(I2$):HCS=""

```

```

4020 GET GR$:GET GR$
4030 PRINT " ";I2
4040 TIMEOUT 0,1,126,I2
4050 RETURN
4052 REM DIGITAL CONTROL
4054 TASK:IF TIMER(125)>0 THEN 4054
4055 IRQ OFF 24
4056 PRINT TS:TASK:PRINT CRS
4057 GET GR$:GET GR$:GET GR$
4060 PRINT"CONTROL SWITCHES:":PRINT"PRESS 0=OFF,1=ON, <RETURN>=SKIP"
4070 PRINT:PC=0
4080 IF PC>7 THEN 4130
4090 PRINT OP$(PC);"=";(DIN(22)AND BT(PC))/BT(PC);":":TASK
4100 GET P$:IF P$=""THEN TASK:GOTO 4100
4101 PV=VAL(P$):PRINT P$
4102 IF P$=CHR$(13)OR P$<"0"OR PV>1 THEN TASK:GOTO 4120
4110 DOUT 22, (DIN(22)AND NOT BT(PC))OR BT(PC)*PV
4120 PC=PC+1:GOTO 4080
4130 PC=0
4140 IF PC>7 THEN 4190
4150 PRINT OP$(PC+8);"=";(DIN(23)AND BT(PC))/BT(PC);":":TASK
4160 GET P$:IF P$=""THEN TASK:GOTO 4160
4161 PV=VAL(P$):PRINT P$
4162 IF P$=CHR$(13)OR P$<"0"OR PV>1 THEN TASK:GOTO 4180
4170 DOUT 23, (DIN(23)AND NOT BT(PC))OR BT(PC)*PV
4180 PC=PC+1:GOTO 4140
4190 SETTIMER 125,5
4192 GET GR$:TASK:IF TIMER(125)>0 THEN 4192
4193 PRINT RS:SETTIMER 125,5
4194 TASK:IF TIMER(125)>0 THEN 4194
4195 PRINT "    CONT"
4200 TASK:MCS="" :ON COM#1 GOSUB 1230
4210 GOTO 470
4220 RETURN
4222 ITS(IT)="    IE"+CRS+"SAFE AUTOSTART"+CRS+TIS+CRS+DAS+CRS+"$ "
4223 IT=IT+1:MCS="" :GOTO 470
4230 REM START UP
4240 CLR:MCS="M"
4242 CRS=CHR$(13)+" "
4250 A=0:D=0:T=0:GR=0:I=0:J=0:K=0
4260 DIM DD(64),AA(16)
4270 DIM DI(64),AU(16),AL(16),AC(16),AB(16)
4280 DIM AD(16),LA(16),IG(23,5),GC(23)
4290 DIM DNS(64),ANS(16),DP(64),AP(16),TL(80)
4300 DIM DA(64),DC(64),DB(64),LD(64)
4310 DIM ITS(80),OPS(16)
4320 BT(0)=1:BT(1)=2:BT(2)=4:BT(3)=8 ;BT=DIG BIT FOR CONVERSIONS
4330 BT(4)=16:BT(5)=32:BT(6)=64:BT(7)=128
4340 FOR I=0 TO 63
4350 READ DNS(I),DP(I),TL(I),DI(I)
4360 IF DP(I)<24 THEN DC(I)=INT(DP(I)/8):DB(I)=DP(I)AND 7 ;DC&DB=DIG INT CH&BIT
4370 NEXT I:FOR I=0 TO 15
4380 READ ANS(I),AP(I),TL(I+64),AL(I),AU(I)
4390 IF AP(I)<24 THEN AC(I)=INT(AP(I)/8):AB(I)=AP(I)AND 7 ;AC&AB=ANAL INT CH&BIT
4400 NEXT I
4410 FOR I=0 TO 15
4420 READ OP$(I):NEXT
4430 LOGIC 22,0:4,3:LOGIC 22,1:4,5
4440 LOGIC 22,2:4,7:LOGIC 22,3:5,1

```

```

4450 LOGIC 22,4:10,0 BAND NOT 10,1 BAND 10,0
4460 LOGIC 22,5:10,3 BAND NOT 10,4 BAND 10,3
4470 LOGIC 22,6:10,6 BAND NOT 10,7 BAND 10,6
4480 LOGIC 23,0:8,7 BAND NOT 9,0 BAND 8,7
4490 LOGIC 23,1:9,2 BAND NOT 9,3 BAND 9,2
4500 LOGIC 23,3:11,5
4510 TS=CHRS(20):RS=CHRS(18)
4520 DOUT 22,255:DOUT 23,251
4530 GOTO 80
4540 REM PARAMETER LIST
4550 REM DIGITAL LIST
4560 REM NAME,PRI0,TIMER LENGTH,INT'PT VAL
4570 DATA D1,64,100,1,RP1,48,100,0
4580 DATA D3,64,100,1,RP2,48,100,0
4590 DATA D5,64,100,1,COMP,43,100,0
4600 DATA D7,64,100,1,FRIG,43,100,0
4610 DATA D9,64,100,1,DIFF,48,100,0
4620 DATA CUT,48,100,0,AIR,44,100,1
4630 DATA N2,44,100,1,HOOD,64,100,1
4640 DATA GAS,53,100,1,GP7D,64,100,0
4650 DATA 115V,54,100,0,6V,54,100,0
4660 DATA GP2D,64,100,0,GP1D,64,100,0
4670 DATA GP3D,64,100,0,GP4D,64,100,0
4680 DATA GP5D,64,100,0,GP6D,64,100,0
4690 DATA LSB,49,100,0,LST,49,100,0
4700 DATA D27,64,100,1,D28,64,100,1
4710 DATA D29,64,100,1,D30,64,100,1
4720 DATA D31,64,100,1,D32,64,100,1
4730 DATA D33,64,100,1,V130,42,100,1
4740 DATA V13C,47,100,1,V140,40,100,0
4750 DATA V14C,47,100,1,V150,40,100,1
4760 DATA V15C,47,100,1,V160,41,100,0
4770 DATA V16C,45,100,1,D42,64,100,1
4780 DATA V170,41,100,0,V17C,45,100,1
4790 DATA D45,64,100,1,VH10,40,100,1
4800 DATA VH1C,45,100,1,D48,64,100,1
4810 DATA V120,40,100,0,V12C,46,100,1
4820 DATA D51,64,100,1,V110,42,100,1
4830 DATA V11C,46,100,1,D54,64,100,1
4840 DATA V100,41,100,0,V10C,45,100,1
4850 DATA vr5d,64,100,1,th1d,64,100,1
4860 DATA COOL,64,100,1,BPS,64,100,1
4870 DATA SAFE,64,100,0,ALARM,54,100,1
4880 DATA D63,64,100,1,D64,64,100,1
4890 REM ANALOG LIST
4900 REM NAME,PRI0,TIMER,LOWER & UPPER VALUE
4910 DATA TH1,53,100,2858,3080
4920 DATA TH2,53,100,-1.983
4930 DATA VR5,64,100,0,4096
4940 DATA GP1,53,100,-1.2048
4950 DATA GP4,52,100,-1.2048
4960 DATA LSB,64,100,-1.5000
4970 DATA LST,64,100,-1.5000
4980 DATA TC1,64,100,-1.5000
4990 DATA TC2,64,100,-1.5000
5000 DATA A10,64,100,-1.5000
5010 DATA GP2,52,100,-1.410
5020 DATA GP3,52,100,-1.410
5030 DATA GP6,51,100,-1.410

```

5040 DATA GP7,51,100,-1,410
5050 DATA GP5,51,100,-1,5000
5060 DATA A16,64,100,0,4096
5070 DATA RP2,COMP,FRIG,DIFF,VP12,N/U,VP10,VP13
5080 DATA VP16,VP17,VP11,ALRM,SAFE,N/U,N/U,N/U
5090 REM H2 SAFE DATA ACQ.
5100 REM WRITTEN BY RU IGARASHI
5110 REM REVISED 21/08/86
6000 STOP:END

E.2: H2 PC (Apple IIe monitor program listing)

```

10 GOTO6310 ;GOTO START-UP ROUTINE
20 REM INPUT ROUTINE ;INPUT ROUTINE
30 CCS="":M=0
40 PRINT CRS:PRINT DS;"IN#2"
50 VTAB1
60 IF M>5THEN130 ;CHECK FOR USER COMMAND
70 IF PEEK(-16384)<127THENM=M+1:GOTO60 ;CCS=USER COMMAND CODE
80 GETCC$:POKE-16368,0
90 IF CC$="X"THEN6160
100 IF CC$="Q"THEN6223
110 RETURN
130 M=0:L=1
132 POKE-16368,128
140 VTAB23:HTAB38:PRINT " ":VTAB1
150 PRINT CRS:PRINT DS;"PR#2"
160 PRINT CRS:PRINT DS;"IN#2"
170 PRINT"S":PRINT DS;"PR#0" ;TELL SAFE THAT ALL READY
180 INPUTHC$:IF HC$="N"THEN30 ;CHECK ACKNOWLEDGEMENT
190 INPUTDT$(L):IF DT$(L)="$"THEN210 ;INPUT DATA, DT$=DATA WORD
200 L=L+1:GOTO190
210 DT$(1)=RIGHT$(DT$(1),2):ICS=LEFT$(DT$(1),1):TP$=RIGHT$(DT$(1),1)
220 VTAB23:HTAB38:INVERSE:PRINT " ":NORMAL
222 POKE-16368,0
230 IF ICS="P"THEN3960 ;CHECK FOR INTERRUPT
240 IF ICS="Q"THEN3960 ;ICS=INTPT CODE
250 IF ICS<>"D"THENRETURN
260 IF TP$="D"THEN370 ;TP$=DATA TYPE CODE
270 P=MA+S*32
280 FOR M=2 TO L-1:C=M-2 ;DECODE AND STORE DATA
290 DA(C)=VAL(DT$(M)) ;DA=ANALOG DATA
300 POKEP+C*2+1,INT(DA(C)/256)
310 POKEP+C*2,DA(C)-INT(DA(C)/256)*256
320 NEXT:P=0
330 FOR C=2 TO 15:Y0=1+INT(C/4):X0=(C+4-Y0*4)*60 ;PLOT ANALOG DATA
340 XN=X0+30+S:YN=(Y0-DA(C)/4095)*40 ;X0,Y0=LAST COORDS
350 H$PLOT XN,Y0*40 TO XN,YN:NEXT ;XN,YN=NEW COORDS
360 FR=FRE(0):RETURN
370 S=S+1:IF S>19THENGOSUB470 ;MOVE DATA?
380 P=MD+S*8
390 FOR M=2 TO L-1:C=M-2
400 DD(C)=VAL(DT$(M)):POKEP+C,DD(C):NEXT ;DD=DIGITAL RAW DATA
410 FOR M=0 TO 7:B=256
420 FOR P=0 TO 7:B=B/2:C=M*8+7-P
430 IF DD(M)-B<0THEND(C)=0:GOTO450
440 D(C)=1:DD(M)=DD(M)-B ;D=DIGITAL DATA
450 NEXTP:NEXT
460 RETURN
470 FOR P=MA-640 TO MA-1 ;SHIFT DATA
480 POKEP,PEEK(P+640):NEXT
490 FOR P=MD-160 TO MD-1
500 POKEP,PEEK(P+160):NEXT
510 POKE3073,MA-INT(MA/256)*256:POKE3074,INT(MA/256)
520 P=MA+639:POKE3075,P-INT(P/256)*256:POKE3076,INT(P/256)
530 P=MD+576+OX*850:POKE3078,INT(P/256):POKE3077,P-INT(P/256)*256
540 POKE3079,0:CALL3082 ;COPY DATA TO AUX MEMORY
550 POKE3073,MD-INT(MD/256)*256:POKE3074,INT(MD/256)
560 P=MD+159:POKE3075,P-INT(P/256)*256:POKE3076,INT(P/256)

```



```

570 P=25216+OX*850:POKE3077,P-INT(P/256)*256:POKE3078,INT(P/256)
580 POKE3079,0:CALL3082:OX=OX+1
590 IF OX>5THENOX=0:IF TD=1THEN GOSUB5250 ;DISK DUMP?
600 POKE3073,00:POKE3074,64
610 POKE3075,255:POKE3076,95
620 POKE3077,00:POKE3078,32
630 POKE3079,1:CALL3082
640 FOR C=2 TO 15:Y0=1+INT(C/4):X0=(C+4-Y0*4)*60
650 FOR M=-20 TO -1:FR=MA+M*32+C*2
660 DA(C)=PEEK(FR)+PEEK(FR+1)*256
670 XN=X0+30+M:YN=(Y0-DA(C)/4095)*40
680 H PLOT XN,Y0*40 TO XN,YN:NEXTM:NEXT
690 S=0:RETURN
700 REM GRAPH DISPLAY:SET UP
710 TEXT:HOME:MO$="G":HCOLOR=7
720 POKE-16304,0:POKE-16297,0
730 POKE35,1:TP$="":VTAB22
740 INVERSE:PRINT"GRAPH":NORMAL:VTAB24
750 PRINT"X-EXIT PC C-COMMAND N-NUMERIC Q-QUIT"
760 FOR M=-20 TO S-1:FR=MA+M*32
770 DA(0)=PEEK(FR)+PEEK(FR+1)*256
780 DA(1)=PEEK(FR+2)+PEEK(FR+3)*256
790 DA(0)=(DA(0)-2457)/.3:DA(1)=2*DA(1)
800 IF DA(0)<0THENDA(0)=0
810 IF DA(0)>4095THENDA(0)=4095
820 IF DA(1)>4095THENDA(1)=4095
830 XN=30+M:YN=(1-DA(0)/4095)*40
840 H PLOT XN,40 TO XN,YN
850 XN=90+M:YN=(1-DA(1)/4095)*40
860 H PLOT XN,40 TO XN,YN:NEXT
870 DA(0)=0:DA(1)=0:RETURN
880 REM GRAPH DISPLAY ;GRAPH SCREEN
890 GOSUB710
900 IF TP$="D"THEN1000
910 DA(0)=(DA(0)-2457)/.3:IF DA(0)<0THENDA(0)=0
920 IF DA(0)>4095THENDA(0)=4095
930 DA(1)=2*DA(1):IF DA(1)>4095THENDA(1)=4095
940 XN=30+S:YN=(1-DA(0)/4095)*40
950 H PLOT XN,40 TO XN,YN
960 XN=90+S:YN=(1-DA(1)/4095)*40
970 H PLOT XN,40 TO XN,YN
980 IF IC$="I"ORCC$="G"THEN1010
990 GOTO1110
1000 IF S=0THEN GOSUB760
1010 FOR M=0 TO 15:HCOLOR=7*D(M)
1020 H PLOT 248,3+M*4 TO 252,3+M*4
1030 NEXT:FOR M=0 TO 15:HCOLOR=7*D(M+16)
1040 H PLOT 268,3+M*4 TO 272,3+M*4
1050 NEXT
1060 FOR M=16 TO 31:HCOLOR=7*D(M+16)
1070 H PLOT 248,19+M*4 TO 252,19+M*4
1080 NEXT:FOR M=16 TO 31:HCOLOR=7*D(M+32)
1090 H PLOT 268,19+M*4 TO 272,19+M*4
1100 NEXT:HCOLOR=7
1110 GOSUB30
1120 IF IC$<>"D"THEN3960
1130 IF CC$=""THEN900
1140 IF CC$="N"THEN1180
1150 IF CC$="C"THEN3150

```

```

1160 GOTO1110
1170 REM  ALPHAMERIC DISPLAY                                ;ALPHAMERIC SCREEN
1180 TEXT:HOME:POKE35,1:MO$="N"
1190 HOME:POKE35,1:MO$="N"                                ;MO$=SCREEN MODE CODE
1200 VTAB22:INVERSE:PRINT"NUMERIC":NORMAL
1210 VTAB24:PRINT"X-EXIT PC C-CMD G-GRAPH H-H.BAR Q-QUIT"
1220 VTAB4:INVERSE:PRINTSPC(40)
1230 VTAB10:PRINTSPC(40)
1240 FOR M=11 TO 21:VTABM:HTAB20
1250 PRINT" ":NEXT:NORMAL
1260 VTAB2:HTAB10:PRINT"N2";SPC(18);"115V"
1270 VTAB3:HTAB10:PRINT"AIR":HTAB31:PRINT"+6V"
1280 VTAB5:PRINT"RP1  GP6      VP16      VP13  VP14"
1290 VTAB6:HTAB22:PRINT"VP15"
1300 VTAB8:PRINT"RP2  GP7      VP17      VP12  VP11"
1310 VTAB11:PRINT"DIFF      VP10  VH1"
1320 HTAB21:PRINT"FRIG  COMP"
1330 VTAB13:HTAB21:PRINT"VR5"
1340 VTAB14:PRINT"GP3      GP4"
1350 VTAB15:HTAB21:PRINT"TH1      (TGT)"
1360 VTAB16:PRINT"GP2      GP1"
1370 HTAB21:PRINT"TH2      (VB)"
1380 VTAB17:HTAB24:PRINT"TC1 (CH)"
1390 VTAB18:HTAB24:PRINT"TC2 (FL)"
1400 VTAB19:PRINT"GP5":HTAB21:PRINT"LST"
1410 VTAB20:HTAB21:PRINT"LSB"
1420 TP$=""
1430 IF TP$="D"THEN1680
1440 VTAB15:HTAB25:PRINTINT(DA(0)/4.095+.5)/100;"V "
1450 VTAB16:HTAB25:MN=7.8*(DA(1)/409.5-.073)
1460 IF MN<0THENMN=0:GOTO1480
1470 MN=INT(130.7*(MN*.01657)+.5)/10
1480 PRINTMN;" K "
1490 VTAB17:HTAB11:M=3:GOSUB1650:PRINTMN;"E";XP;"T "
1500 VTAB15:HTAB11:M=4:GOSUB1650:PRINTMN;"E";XP;"T "
1510 VTAB20:HTAB28:PRINTINT(DA(5)/4.095+.5)/100;"V "
1520 VTAB19:HTAB28:PRINTINT(DA(6)/4.095+.5)/100;"V "
1530 VTAB17:HTAB32:PRINTINT(DA(7)/4.095+.5)/100;"V "
1540 VTAB18:HTAB32:PRINTINT(DA(8)/4.095+.5)/100;"V "
1550 VTAB17:M=10:GOSUB1620:PRINTMN;"E";XP;"MB "
1560 VTAB15:M=11:GOSUB1620:PRINTMN;"E";XP;"MB "
1570 VTAB6:HTAB6:M=12:GOSUB1620:PRINTMN;"E";XP;"MB "
1580 VTAB9:HTAB6:M=13:GOSUB1620:PRINTMN;"E";XP;"MB "
1590 VTAB20:M=14:GOSUB1620:PRINTMN;"E";XP;"MB "
1600 IF CC$="N"ORIC$="I"ORIC$="A"THEN1680
1610 GOTO2410
1620 MN=.496*(DA(M)/409.5)-2.827-.25*SIN(.669*DA(M)/409.5)
1630 XP=INT(MN):MN=INT(10^(MN-XP)*80)/100
1640 RETURN
1650 MN=.536*(DA(M)/409.5)-6.941-.393*SIN(.510*DA(M)/409.5)-
1660 XP=INT(MN):MN=INT(10^(MN-XP)*80)/100
1670 RETURN
1680 IF D(61)=0THENFLASH
1690 VTAB2:PRINT"ALARM":NORMAL:VTAB3
1700 IF D(14)=0THENPRINT"GAS OK":GOTO1720
1710 FLASH:PRINT"H2 LEAK";GS;GS;GS:NORMAL
1720 VTAB6:HTAB17:M=39:GOSUB1820
1730 VTAB9:HTAB18:M=42:GOSUB1820
1740 VTAB7:HTAB23:M=37:GOSUB1820

```

```
1750 VTAB6:HTAB30:M=33:GOSUB1820
1760 HTAB35:M=35:GOSUB1820
1770 VTAB9:HTAB30:M=48:GOSUB1820
1780 HTAB36:M=51:GOSUB1820
1790 VTAB12:HTAB11:M=54:GOSUB1820
1800 HTAB16:M=45:GOSUB1820
1810 GOTO1850
1820 IF D(M)=D(M+1)THENFLASH:PRINT"O C":NORMAL:RETURN
1830 IF D(M)=1THENINVERSE:PRINT"O";:NORMAL:PRINT" C":RETURN
1840 PRINT"O ";:INVERSE:PRINT"C":NORMAL:RETURN
1850 VTAB2:HTAB14:M=12:GOSUB1880
1860 VTAB3:HTAB14:M=11:GOSUB1880
1870 GOTO1900
1880 IF D(M)=0THENPRINT"OK":RETURN
1890 FLASH:PRINT"LO":NORMAL:RETURN
1900 VTAB5:HTAB10:M=23:GOSUB1980
1910 VTAB8:HTAB10:M=15:GOSUB1980
1920 VTAB16:HTAB15:M=19:GOSUB1980
1930 HTAB5:M=18:GOSUB1980
1940 VTAB14:HTAB5:M=20:GOSUB1980
1950 HTAB15:M=21:GOSUB1980
1960 VTAB19:HTAB5:M=22:GOSUB1980
1970 GOTO2000
1980 IF D(M)=0THENPRINT"OK":RETURN
1990 FLASH:PRINT"NO":NORMAL:RETURN
2000 VTAB2:HTAB35:M=16:GOSUB2110
2010 VTAB3:HTAB35:M=17:GOSUB2110
2030 HTAB20:M=13:REMGOSUB2110
2040 VTAB6:HTAB3:M=1:GOSUB2110
2050 VTAB9:HTAB3:M=3:GOSUB2110
2060 VTAB12:HTAB3:M=9:GOSUB2110
2070 HTAB23:M=7:GOSUB2110
2080 HTAB29:M=5:GOSUB2110
2084 VTAB19:HTAB25:M=25:GOSUB2110
2090 VTAB20:HTAB25:M=24:GOSUB2110
2100 GOTO2160
2110 IF D(M)=1THENPRINT"ON":RETURN
2120 FLASH:PRINT"NO":NORMAL:RETURN
2160 VTAB6:M=0:GOSUB2280
2170 HTAB28:M=32:GOSUB2280
2180 VTAB9:M=2:GOSUB2280
2190 HTAB16:M=41:GOSUB2280
2200 HTAB28:M=47:GOSUB2280
2210 HTAB34:M=50:GOSUB2280
2220 VTAB12:M=8:GOSUB2280
2230 HTAB10:M=53:GOSUB2280
2240 HTAB15:M=44:GOSUB2280
2250 HTAB21:M=6:GOSUB2280
2260 HTAB27:M=4:GOSUB2280
2270 GOTO2300
2280 IF D(M)=0THENPRINT"A":RETURN
2290 INVERSE:PRINT"M":NORMAL:RETURN
2300 VTAB12:HTAB6:PRINT" "
2310 IF D(10)=1THENHTAB6:PRINT"CUT"
2320 VTAB11:HTAB33:IF D(58)<>D(59)THEN2340
2330 FLASH:PRINT"COOL":VTAB12:HTAB33:PRINT"BYPASS":NORMAL:GOTO2370
2340 IF D(58)=1THENPRINT"COOL":INVERSE:GOTO2360
2350 INVERSE:PRINT"COOL":NORMAL
2360 VTAB12:HTAB33:PRINT"BYPASS":NORMAL
```

```

2370 VTAB13:HTAB26:IF D(56)=1THENINVERSE
2380 PRINT"LOCAL":NORMAL
2390 HTAB33:IF D(57)=0THENPRINT"TH1,";:INVERSE:PRINT"TH2":NORMAL:GOTO2
2400 INVERSE:PRINT"TH1";:NORMAL:PRINT",TH2"
2410 GOSUB30
2420 IF IC$<>"D"THEN3960
2430 IF CC$=""THEN1430
2440 IF CC$="G"THEN890
2450 IF CC$="C"THEN3150
2460 IF CC$="H"THEN2490
2470 GOTO2410
2480 REM HORIZONTAL BAR
2490 TEXT:HOME:POKE35,1:MOS="H"
2500 VTAB24:PRINT"X-EXIT PC C-COMMAND N-NUMERIC Q-QUIT"
2510 VTAB3:PRINTNA$(0):VTAB4:HTAB9
2520 PRINT"  15.5^ ^16 ^17  PSIA"
2530 VTAB5:PRINTNA$(1):VTAB6:HTAB9
2540 PRINT" ^15 ^18 ^20 ^22 ^23 K"
2550 VTAB7:PRINTNA$(5):VTAB8:PRINTNA$(6)
2560 VTAB9:HTAB9
2570 PRINT"OV";SPC(12);"^5V";SPC(12);"10V"
2580 VTAB10:PRINTNA$(7):VTAB11:PRINTNA$(8)
2590 VTAB12:HTAB9
2600 PRINT"OV";SPC(12);"^5V";SPC(12);"10V"
2610 VTAB13:PRINTNA$(3):VTAB14:PRINTNA$(4)
2620 VTAB15
2630 PRINT"  10E ^-7 ^-6 ^-5 ^-4 ^-3 T"
2640 VTAB16:PRINTNA$(10):VTAB17:PRINTNA$(11)
2650 VTAB18
2660 PRINT"  10E -3^ ^-2 ^-1 ^0 ^1 ^3"
2670 VTAB19:PRINTNA$(12):VTAB20:PRINTNA$(13)
2680 VTAB21:PRINTNA$(14)
2690 TPS=""
2700 IF TPS="D"THEN2870
2710 DA(0)=(DA(0)-2457)/.3
2720 DA(1)=2*DA(1)
2730 FOR P=0 TO 15
2740 HH(P)=INT(DA(P)/4095*30+.5):NEXT
2750 IF HH(0)>30THENHH(0)=30
2760 IF HH(0)<0THENHH(0)=0
2770 IF HH(1)>30THENHH(1)=30
2780 VTAB3:P=0:GOSUB2850:VTAB5:P=1:GOSUB2850
2790 VTAB7:P=5:GOSUB2850:VTAB8:P=6:GOSUB2850
2800 VTAB10:P=7:GOSUB2850:VTAB11:P=8:GOSUB2850
2810 VTAB13:P=3:GOSUB2850:VTAB14:P=4:GOSUB2850
2820 VTAB16:P=10:GOSUB2850:VTAB17:P=11:GOSUB2850
2830 VTAB19:P=12:GOSUB2850:VTAB20:P=13:GOSUB2850
2840 VTAB21:P=14:GOSUB2850:GOTO2870
2850 HTAB9:INVERSE:PRINTSPC(HH(P));
2860 NORMAL:PRINTSPC(30-HH(P)):RETURN
2870 GOSUB30:IF IC$<>"D"THEN3960
2880 IF CC$=""THEN2700
2890 IF CC$="C"THEN3150
2900 IF CC$="N"THEN1180
2910 GOTO2870
2920 REM RAW DATA MODE
2930 TEXT:HOME:POKE35,1:MOS="R"
2940 VTAB22:INVERSE:PRINT"RAW DATA":NORMAL
2950 VTAB24:PRINT"X-EXIT C-CMD N-NUMERIC H-H.BAR Q-QUIT"

```

HORIZONTAL BAR SCREEN

;HH=H. BAR DUMMY DATA VARIABLE

;RAW DATA SCREEN


```

2960 TP$=""
2970 IF TP$="D"THEN3030
2980 FOR M=0 TO 15:VTAB(3+M)
2990 PRINTNA$(M);"=" ;INT(DA(M)/4.095+.5)/100;" V";
3000 IF AE$(M)="0"THENPRINT"D ";
3010 PRINT:NEXT
3020 IF CC$<>"R"ANDICS$<>"I"THEN3070
3030 FOR M=0 TO 63:VTAB3+M-INT(M/16)*16
3040 HTAB14+INT(M/16)*6:PRINTND$(M);
3050 PRINTD(M);:IF DE$(M)="0"THENPRINT"D"
3060 NEXT
3070 GOSUB30
3080 IF IC$<>"D"THEN3960
3090 IF CC$=""THEN2970
3100 IF CC$="C"THEN3150
3110 IF CC$="N"THEN1180
3120 IF CC$="H"THEN2490
3130 GOTO3070
3140 REM COMMAND MODE ;COMMAND SCREEN
3150 TEXT:HOME:MO$="C"
3160 POKE35,21:HOME:VTAB22
3170 INVERSE:PRINT"COMMAND":NORMAL:VTAB2
3180 PRINT"X-EXIT PC, LEAVE SAFE RUNNING"
3190 PRINT"G-Graphics":PRINT"N-NUMERIC"
3200 PRINT"H-HORIZ. BAR GRAPH":PRINT"R-RAW DATA"
3210 PRINT"M-MODIFY PARAMETER":PRINT"S-SET TIME&/DATE"
3220 PRINT"Q-QUIT":PRINT"P-PRINT DUMP"
3230 PRINT"T-TOGGLE PRINTER ENABLE (FROM ";PT;" )"
3240 PRINT"I-SET PRINTER DUMP INTERVAL"
3250 PRINT"D-TOGGLE DATA DISK (FROM ";TD;" )"
3254 PRINT"F-DISK FILE DIRECTORY"
3260 PRINT"A-TOGGLE ALL AVAILABLE ALARMS (FROM ";AM;" )"
3270 PRINT"O-DIGITAL OUTPUT CONTROL"
3280 PRINT"B-TOGGLE 'BEEP' (FROM ";ASC(G$)/7;" )"
3290 POKE35,1
3300 GOSUB30
3310 IF IC$<>"D"THEN3960
3320 IF CC$=""THEN3300
3330 IF CC$="G"THEN890
3340 IF CC$="N"THEN1180
3350 IF CC$="H"THEN2490
3360 IF CC$="R"THEN2930
3370 IF CC$="M"THEN3470
3380 IF CC$="P"THENGOSUB4430
3390 IF CC$="T"THEN3690
3400 IF CC$="D"THEN5570
3402 IF CC$="F"THEN5552
3410 IF CC$="A"THEN3740
3420 IF CC$="B"THEN3800
3430 IF CC$="I"THEN3840
3440 IF CC$="S"THEN3590
3450 IF CC$="O"THEN3640
3460 GOTO3300
3470 POKE35,21:HOME
3480 PRINT CR$:PRINT D$;"PR#2"
3490 PRINT CR$:PRINT D$;"IN#2"
3500 PRINT"M";:PRINT CR$:PRINT D$;"PR#0":STOP
3510 GOSUB3520:GOTO3160
3520 P=0:HOME:POKE35,1:VTAB1

```



```

3530 INPUTDES(P):IF DES(P)="$"THENP=0:GOTO3550      ;DES=DIG ALARM ENABLE STATUS
3540 P=P+1:GOTO3530
3550 INPUTAES(P):IF AES(P)="$"THEN3570              ;AES=ANAL ALARM ENABLE STATUS
3560 P=P+1:GOTO3550
3570 PRINT CRS:PRINT DS;"IN#0"
3580 RETURN
3590 POKE35,21:HOME
3600 PRINT CRS:PRINT DS;"PR#2"
3610 PRINT CRS:PRINT DS;"IN#2"
3620 PRINT"TR";:PRINT CRS:PRINT DS;"PR#0":STOP
3630 GOTO3150
3640 POKE35,24:HOME
3650 PRINT CRS:PRINT DS;"PR#2"
3660 PRINT CRS:PRINT DS;"IN#2"
3670 PRINT"OR";:PRINT CRS:PRINT DS;"PR#0":STOP
3680 GOTO3150
3690 IF PT=1THENPT=0:GOTO3730
3700 PT=1:VTAB1
3710 PRINT CRS:PRINT DS;"PR#1"
3720 PRINTES;"J,60,960,$":PRINT DS;"PR#0"
3730 VTAB23:PRINT"PRINTER ENABLE TOGGLED TO ";PT;BLS:GOTO3160
3740 IF AM=1THENAM=0:GOTO3760
3750 AM=1
3760 PRINT CRS:PRINT DS;"PR#2":PRINT"A"
3770 PRINT CRS:PRINT DS;"PR#0":PRINT CRS
3780 PRINT DS;"IN#2":INPUTAMS:AM=VAL(AMS)
3790 VTAB23:PRINT"ALL ALARMS SET TO ";AM;BLS:GOTO3160
3800 VTAB23:PRINT"'BEEP' ENABLE TOGGLED ";
3810 IF GS=""THENGS=CHRS(7):PRINT"ON":GOTO3830
3820 GS="" :PRINT"OFF";BLS
3830 GOTO3160
3840 POKE34,18:POKE35,20:HOME
3850 INPUT"PRINTER DUMP INTERVAL = ? (MIN) ";NI
3860 PRINT"DUMP EVERY ";NI;"MIN ? (Y/N)"
3870 GETYNS:PRINTYNS:IF YNS="N"THEN3850
3880 IF YNS<>"Y"THEN3870
3890 PRINT CRS:PRINT DS;"PR#2":PRINT"I"
3900 PRINTINT(NI*4);"$"
3910 PRINT CRS:PRINT DS;"PR#0":PRINT CRS
3920 PRINT DS;"IN#2":INPUTNI:VTAB23
3930 PRINT"SAFE 8000 HAS SET TO ";NI/4;"MIN";BLS
3940 POKE34,0:GOTO3160
3950 REM   INTERRUPT ROUTINE                      ;INTERRUPT SCREEN
3960 N=L-2:M=L-1:C=0
3970 DYS=RIGHT$(DT$(M),2)+"/"+MID$(DT$(M),3,2)+"/"+LEFT$(DT$(M),2)      ;DYS=DATE
3980 TMS=LEFT$(DT$(N),2)+":"+MID$(DT$(N),3,2)+":"+RIGHT$(DT$(N),2)      ;TMS=TIME
3990 IF ICS="P"THEN4030
4000 IF ICS="Q"THEN6230
4002 IF TPS="E"THEN4352
4010 NMS=DT$(2):C=VAL(DT$(3)):IN$(I)=NMS
4020 IF TD=0THEN4070
4030 ID(I,1)=C:ID(I,2)=VAL(DT$(4))      ;ID=INTPT DATA STOAGE FOR
4040 ID(I,3)=VAL(DT$(N)):ID(I,4)=VAL(DT$(M))      ; DISK DUMP
4050 ID(I,0)=0:IF TPS="A"THENID(I,0)=1
4060 I=I+1
4070 VTAB23:PRINTBLS;BLS:VTAB22
4080 HTAB20:PRINTTMS;" ";DYS:VTAB23
4090 IF TPS="D"THEN4230
4100 DA(C)=VAL(DT$(4))

```

```

4110 PRINTNMS;"=";INT(DA(C)/.4095+.5)/1000;
4120 IN$="V *SET PT EXCEEDED*"
4130 IF IC$<>"L"THEN4160
4140 IN$="V *2ND ALARM*":FLASH
4150 FOR K=1 TO 3:PRINTGS;:NEXTK
4160 PRINTIN$;BL$
4170 NORMAL:IF PT=0THEN4210                ;OUTPUT TO PRINTER?
4180 VTAB1:PRINT CR$:PRINT DS;"PR#1":POKE33,33
4190 PRINTTMS;" ";DY$;" ";NMS;"=";DA(C)/409.5;IN$
4200 POKE33,40:PRINT DS;"PR#0"
4210 IF MO$="N"THENIC$="D":GOTO2410
4220 POKE35,21:GOTO1190
4230 M=INT(C/8):P=C-M*8:DD(M)=VAL(DTS(4))
4240 D(C)=INT(DD(M)/2^P):D(C)=D(C)-2*INT(D(C)/2)
4250 PRINT"SWITCH:";NMS;
4260 IN$=" ALARM":IF IC$<>"L"THEN4290
4270 IN$=" *2ND ALARM*":FLASH
4280 FOR K=1 TO 3:PRINTGS;:NEXTK
4290 PRINTIN$;BL$:NORMAL
4300 IF PT=0THEN4340
4310 VTAB1:PRINT CR$:PRINT DS;"PR#1":POKE33,33
4320 PRINTTMS;" ";DY$;" ";NMS;"=";D(C);IN$
4330 POKE33,40:PRINT DS;"PR#0"
4340 IF MO$="N"THENIC$="D":GOTO2410
4350 POKE35,21:GOTO1190
4351 REM SAFE AUTOSTART O/P
4352 VTAB22:HTAB20:PRINTTMS;" ";DY$:VTAB23
4353 FLASH:PRINTDTS(2):NORMAL:VTAB1
4354 RETURN
4360 REM PRINT DUMP                        ;PRINTER DUMP OF DATA
4370 VTAB22:HTAB20:PRINTTMS;" ";DY$
4380 VTAB23:INVERSE:PRINT"PRINT DUMP IN PROGRESS"
4390 NORMAL:VTAB1
4400 REM IF TD=1 THEN GOSUB 4950
4410 FR=FRE(0)
4420 IF PT=0THEN5200
4430 PRINT CR$:PRINT DS;"PR#1"
4440 PRINTES;"F,360,660,900,$"
4450 POKE33,33:PRINTTMS,DY$
4460 M=0:GOSUB4720:PRINTIS;
4470 PRINT" ";IS;
4480 M=7:GOSUB4720:PRINTIS;
4490 M=6:GOSUB4720:IF D(25)=0THENPRINT"OFF":GOTO4510
4500 PRINT"ON"
4510 M=1:GOSUB4720:MN=7.8*(DA(1)/409.5-.073)
4520 IF MN<0THENMN=0
4530 PRINT"=";INT(130.7*(MN^0.1657)+.5)/10;"K";IS;
4540 M=8:GOSUB4720:PRINTIS;
4550 M=5:GOSUB4720:IF D(24)=0THENPRINT"OFF":GOTO4570
4560 PRINT"ON"
4570 M=3:GOSUB4720:GOSUB4770
4580 M=19:GOSUB4800:PRINTIS;
4590 M=10:GOSUB4720:GOSUB4740
4600 M=18:GOSUB4800:PRINT
4610 M=4:GOSUB4720:GOSUB4770
4620 M=21:GOSUB4800:PRINTIS;
4630 M=11:GOSUB4720:GOSUB4740
4640 M=20:GOSUB4800:PRINT
4650 M=12:GOSUB4720:GOSUB4740

```

```

4660 M=23:GOSUB4800:PRINT$;
4670 M=13:GOSUB4720:GOSUB4740
4680 M=15:GOSUB4800:PRINT
4690 M=14:GOSUB4720:GOSUB4740
4700 M=22:GOSUB4800:PRINT
4710 GOTO4820
4720 PRINTN$(M);"=";INT(DA(M)/4.095+.5)/100;"V ";
4730 RETURN
4740 MN=.4088*DA(M)/409.5-2.428-.96*EXP(-5*DA(M)/409.5)
4750 XP=INT(MN):MN=INT(10^(MN-XP)*80)/100
4760 PRINT"=";MN;"E";XP;"MBAR ";:RETURN
4770 MN=.435*DA(M)/409.5-6.259-.75*EXP(-DA(M)/1.18/409.5)
4780 XP=INT(MN):MN=INT(10^(MN-XP)*80)/100
4790 PRINT"=";MN;"E";XP;"TORR ";:RETURN
4800 IF D(M) THENPRINT"NO";:RETURN
4810 PRINT"OK";:RETURN
4820 PRINT$;"F,160,260,360,460,560,660,760,860,$"
4830 PRINT"RP1-";:M=1:GOSUB5180
4840 PRINT"VP16-";:M=39:GOSUB5150
4850 PRINT"VP15-";:M=37:GOSUB5150
4860 PRINT"VP13-";:M=33:GOSUB5150
4870 PRINT"VP14-";:M=35:GOSUB5150
4880 PRINT"DIFF-";:M=9:GOSUB5180
4890 PRINT"VP10-";:M=54:GOSUB5150:PRINT
4900 PRINT"RP2-";:M=3:GOSUB5180
4910 PRINT"VP17-";:M=42:GOSUB5150
4920 PRINT$;"VP12-";:M=48:GOSUB5150
4930 PRINT"VP11-";:M=51:GOSUB5150
4940 IF D(10)=1THENPRINT$;:GOTO4960
4950 PRINT"CUTOUT";IS;
4960 PRINT"VH1-";:M=45:GOSUB5150:PRINT
4970 PRINT"FRIG-";:M=7:GOSUB5180
4980 PRINT"COMP-";:M=5:GOSUB5180
4990 IF D(57) THENPRINT"TH1";:GOTO5010
5000 PRINT"TH2";
5010 PRINT"CONTROL"
5140 PRINT:GOTO5200
5150 IF D(M)=D(M+1) THENPRINT"T";IS;:RETURN
5160 IF D(M) THENPRINT"O";IS;:RETURN
5170 PRINT"C";IS;:RETURN
5180 IF D(M) THENPRINT"ON";IS;:RETURN
5190 PRINT"OFF";IS;:RETURN
5200 PRINT D$;"PR#0"
5210 VTAB23:PRINT$PC(22):POKE33,40
5220 IC$="":IF CC$<>"P" THEN30
5230 CC$="":RETURN
5240 REM SAVE DATA ON DISK ;DISK DUMP OF DATA
5242 VTAB23:INVERSE:PRINT"DISK DUMP IN PROGRESS":NORMAL:VTAB1
5250 PRINT D$;"OPEN ";DR$;"L128" ;DR$=DATA FILENAME
5260 DC=DC+1:PRINT D$;"WRITE ";DR$;"R";DC
5270 PRINT-9;CR$;TMS;CR$;DYS:DC=DC+1 ;DC=FILE DATA COUNTER
5280 FOR C=0 TO OX-1:P=24576+C*850
5290 POKE3073,P-INT(P/256)*256:POKE3074,INT(P/256)
5300 P=P+799:POKE3075,P-INT(P/256)*256:POKE3076,INT(P/256)
5310 POKE3077,MA-INT(MA/256)*256:POKE3078,INT(MA/256)
5320 POKE3079,1:CALL3082
5330 FOR M=0 TO 1
5340 PRINT D$;"WRITE ";DR$;"R";DC+M
5350 FOR P=0 TO 15:FR=MA+M*10*32+P*2

```

```

5360 PRINTPEEK(FR)+PEEK(FR+1)*256
5370 NEXTP:PRINT-7
5380 FOR P=0 TO 7
5390 PRINTPEEK(MD-160+M*10*8+P)
5400 NEXTP:PRINT-8:NEXTM
5410 DC=DC+M-1:NEXTC:OX=0:TB=0
5420 PRINT D$;"OPEN ";IR$;"",L32" ;IR$=INTERRUPT FILENAME
5430 FOR M=0 TO I-1
5440 PRINT D$;"WRITE ";IR$;"",R";IC+M ;IC=INTPT COUNTER
5450 FOR P=0 TO 4
5460 PRINTID(M,P):NEXTP:PRINTINS(M):NEXTM
5470 PRINT D$;"CLOSE ":IC=IC+I
5480 P=2:PRINT D$;"OPEN DIR,L50" ;UPDATE DIRECTORY FILE
5490 PRINT D$;"READ DIR,R";P
5500 INPUTDF$,FR,IR:TB=TB+FR/2+IR/8+2.2
5502 IF DF$<>DR$THENP=P+1:GOTO5490
5510 INPUTOC,NT$
5520 PRINT D$;"WRITE DIR,R";P
5530 PRINTDR$:PRINTDC:PRINTIC:PRINTOC:PRINTNT$ ;DR$=DATA FILENAME,DC=#RECORDS
5540 PRINT D$;"CLOSE" ;IC=#INTPT,OC=OPEN/CLOSE
5542 IF TB<490THEN5550
5543 VTAB23:FLASH:PRINT"DISK NEARLY FULL":NORMAL:VTAB1
5544 PRINTG$;G$;G$:TD=0
5550 S=0:I=0:RETURN
5552 POKE35,21:HOME
5553 PRINT:PRINT D$;"CATALOG":PRINT"HIT ANY KEY TO RETURN TO MENU"
5554 GETYN$:GOTO3160
5560 REM TOGGLE DATA DISK ;TOGGLE DISK DUMP ROUTINE
5570 IF TD=1THENTD=0:GOTO6100
5580 TD=1:POKE34,18:POKE35,21:HOME
5590 OX=0:PRINT CR$
5600 PRINT D$;"OPEN DIR,L50":M=1
5610 ONERRGOTO6130
5620 PRINT D$;"READ DIR,R0"
5630 INPUTDR$:IF DR$="L50"THEN5690 ;NEW DISK?
5640 PRINT D$;"WRITE DIR,R0":PRINT"L50"
5650 PRINT D$;"WRITE DIR,R1"
5660 PRINT"DIR":PRINT0:PRINT0:PRINT1:PRINT"DIRECTORY"
5670 PRINT D$;"WRITE DIR,R2":PRINT"$"
5680 GOTO5610
5690 POKE216,0
5700 PRINT D$;"READ DIR,R";M
5710 INPUTDR$:IF DR$="$"THEN5840 ;INPUT DIRECTORY ENTRY
5720 INPUTDC,IC,OC,NT$:PRINTD$
5730 IF OC=1THENM=M+1:GOTO5690
5740 PRINT"DUMP DATA TO DISK FILE ";DR$
5750 PRINT"OK? ";:GETYN$:PRINTYN$
5760 IF YN$="Y"THEN5980
5770 IF YN$<>"N"THEN6090
5780 PRINT D$;"WRITE DIR,R";M
5790 PRINTDR$:PRINTDC:PRINTIC:PRINT1:PRINTNT$
5800 M=M+1:PRINT D$;"WRITE DIR,R";M
5810 PRINT"$"
5820 PRINT D$;"LOCK ";DR$:PRINT D$;"LOCK ";DR$+"."I" ;LOCK OLD FILES
5830 GOTO5690
5840 PRINT"NO OLD FILES OPEN."
5850 PRINTD$
5860 INPUT"NAME OF NEW FILE? ";DR$
5862 IF DR$=CR$THEN6090

```



```

5870 PRINT"DUMP DATA TO DISK FILE ";DR$
5880 PRINT"OK? ";:GETYNS:PRINTYNS
5890 IF YNS="N"THEN5850
5900 IF YNS<>"Y"THEN6090
5910 P=1
5920 IF P>=MTHENPRINTD$:GOTO5970
5930 PRINT D$;"READ DIR,R";P
5940 INPUTD$:IF D$<>DR$THENP=P+1:GOTO5920
5950 PRINT"THERE IS A PREVIOUS FILE CALLED ";DR$:GOTO5850
5960 PRINT:PRINTNT$
5970 INPUT"ENTER NOTE FOR THIS FILE: ";NT$
5972 DC=0:IC=0
5980 IRS=DR$+".I"
5990 PRINT D$;"WRITE DIR,R";M
6000 PRINTDR$:PRINTDC:PRINTIC:PRINT0:PRINTNT$
6002 PRINT D$;"WRITE DIR,R";M+1
6004 PRINT"$"
6010 PRINT D$;"CLOSE"
6020 PRINT D$;"OPEN ";DR$;",L128" ;OPEN NEW DATA FILE
6040 PRINT D$;"WRITE ";DR$;",R0"
6050 PRINT"L128 DELIM=-9 1ST.REC-TIME&DATE":PRINT"$"
6060 PRINT D$;"CLOSE"
6070 PRINT D$;"OPEN ";IRS$;",L32" ;OPEN NEW INTPT FILE
6080 PRINT D$;"WRITE ";IRS$;",R0":PRINT"L32":PRINT"$"
6090 PRINT D$;"CLOSE"
6100 VTAB23:PRINT"DISK DRIVE TOGGLED TO ";TD;BL$
6110 POKE34,0:GOTO3160
6120 REM RESET ERR AFTER NEW DISK
6130 POKE216,0:PRINT D$;"OPEN DIR,L50"
6140 PRINT D$;"WRITE DIR,R0":PRINT"$"
6150 PRINT D$;"READ DIR,R0":RESUME
6160 PRINT CRS:PRINT D$;"PR#2":PRINT CRS:PRINT D$;"IN#2"
6170 PRINT"X":PRINT D$;"PR#0"
6180 INPUTHCS:IF HCS<>"0"THEN6160
6190 POKE35,24:TEXT
6200 PRINT"PC SHUTDOWN, SAFE 8000 RUN ON MANUAL"
6210 PRINT D$;"IN#0":IF TD=1THENGOSUB5250
6220 END
6223 POKE34,18:POKE35,21:VTAB21
6224 PRINT"ARE YOU SURE YOU WANT TO SHUT":PRINT"EVERYTHING DOWN?"
6225 GETYNS:IF YNS="N"THEN6229
6226 IF YNS<>"Y"THEN6225
6227 PRINT CRS:PRINT D$;"PR#2"
6228 PRINT"Q":PRINT D$;"PR#0"
6229 POKE34,0:POKE35,1:VTAB1:CCS="N":RETURN
6230 TEXT:VTAB24:PRINT"SYSTEM SHUTDOWN AT ";TMS;" ";DTS
6240 IF TD=1THENGOSUB5250
6250 END
6260 REM H2 PC
6270 REM *SAFE 8000 DATA ACQ SYS
6280 REM *WRITTEN BY RU IGARASHI
6290 REM REVISED 28/07/86
6300 REM START UP.
6310 PRINTFRE(0):HOME
6320 L=0:M=0:P=0:C=0:B=0
6330 D$=CHR$(4):CRS=CHR$(13):GS=CHR$(7)
6340 DIMD(64),DTS(40),DT(16),DA(16),DD(16),NAS(16)
6350 DIMHH(16),HO(16),DES(64),AES(16),NDS(64)
6360 DIMID(20,5),INS(20)

```



```

6370 BL$=" " :HC$=""
6380 S=0:I=0:IC$="I":TP$="D"
6390 MN=0:XP=0
6400 XN=0:YN=0:X0=0:Y0=0
6410 MA=6784:MD=7584:OX=0
6420 DR$="DATA":IR$=DR$+" :I":DC=0:IC=0
6430 FOR P=0 TO 63:READNDS(P):NEXT ;DIG DATA PT NAMES
6440 FOR P=0 TO 15:READNAS(P):NEXT ;ANAL DATA PT NAMES
6450 ES=CHRS(27):IS=CHRS(9)
6460 AM$="M":FR=FRE(0)
6470 PRINT CR$:PRINT DS;"PR#2":PRINT CR$:PRINT DS;"IN#2"
6480 PRINT"R":PRINT DS;"PR#0" ;SAFE RUNNING OR NOT?
6490 INPUTHC$:IF HC$<>"X"THEN6520
6500 PRINT CR$:PRINT DS;"PR#2":PRINT"M":PRINT DS;"PR#0" ;COMMAND MODIFY PARAM
6510 GOTO6530
6520 PRINT CR$:PRINT DS;"PR#2":PRINT"RUN 50":PRINT DS;"PR#0"
6530 PRINT CR$:PRINT DS;"IN#2":INPUTYNS:STOP
6540 GOSUB3520
6550 PRINT CR$:PRINT DS;"PR#2"
6560 PRINT CR$:PRINT DS;"IN#2"
6570 POKE35,24:VTAB5
6580 PRINT"T":PRINT DS;"PR#0":STOP ;COMMAND TIME CHANGE
6590 PRINT CR$:PRINT DS;"IN#0"
6600 FOR I=1 TO 46:READM:POKE3081+I,M:NEXT ;STORE DATA TRANSF ROUTINE
6610 I=0
6620 POKE3073,00:POKE3074,64
6630 POKE3075,255:POKE3076,95
6640 POKE3077,00:POKE3078,32
6650 POKE3079,1:CALL3082
6660 PRINT"PUT DATA DISK INTO DISK DRIVE"
6670 PRINT:PRINT"READY FOR MAIN PROGRAM? (Y/N) "
6680 GETYNS:IF YNS="Y"THEN1180
6690 GOTO6680
6700 DATA 1-,2-,3-,4-,5-,6-,7-,8-
6710 DATA 9-,10-,11-,12-,13-,14-,15-,16-
6720 DATA 17-,18-,19-,20-,21-,22-,23-,24-
6730 DATA 25-,26-,27-,28-,29-,30-,31-,32-
6740 DATA 1-,2-,3-,4-,5-,6-,7-,8-
6750 DATA 9-,10-,11-,12-,13-,14-,15-,16-
6760 DATA 17-,18-,19-,20-,21-,22-,23-,24-
6770 DATA 25-,26-,27-,28-,29-,30-,31-,32-
6780 DATA TH1,TH2,VR5,GP1,GP4,LSB,LST,TC1
6790 DATA TC2,N/U,GP2,GP3,GP6,GP7,GP5,N/U
6800 DATA 173,1,12,133,60,173,2,12
6810 DATA 133,61,173,3,12,133,62,173
6820 DATA 4,12,133,63,173,5,12,133
6830 DATA 66,173,6,12,133,67,173,7
6840 DATA 12,201,1,240,4,56,76,52
6850 DATA 12,24,32,17,195,96

```

E.3: H2 PC BOOT (Apple IIe monitor boot program listing)

```

90 DS=CHRS(4)
97 REM SET BOTTOM-OF-BASIC POINTER
98 REM 103=LO BYTE, 104=HI BYTE
100 POKE 103,1:POKE 104,64
107 REM FIRST LOCATION MUST ALWAYS BE 0
110 POKE 16384,0
150 PRINTDS;"RUN H2 PC SET"
200 END

```

E.4: H2 PC SET (Apple IIe monitor set-up program listing - called by H2 PC BOOT)

```

90 REM STORE CHARACTER SET IN MEMORY
100 FT=2048
110 RP=142:I=0
180 HI=INT(FT/256):LO=FT-HI*256
190 POKE 232,LO:POKE 233,HI
200 READNC
210 POKE FT,NC:POKE FT+1,0
220 GOSUB600
300 READCH:IFCH<0THEN1000
310 POKE FT+RP,CH:RP=RP+1
320 IFCH=0THENGOSUB600
330 GOTO300
600 HI=INT(RP/256):LO=RP-HI*256
610 I=I+1:OP=FT+I*2
620 POKE OP,LO:POKE OP+1,HI
630 RETURN
990 REM PUT DATA TRANSFER ROUTINE IN MEMORY
1000 FOR I=1 TO 46:READM
1010 POKE 3081+I,M:NEXT
1020 FOR I=6144 TO 7744:POKE I,0:NEXT
1190 REM DRAW UP AND STORE EMPTY GRAPH SCREEN
1200 HGR:TEXT:HCOLOR=7
1210 HPLOT 0,0 TO 0,159:HPLOT TO 279,159
1220 HPLOT 0,40TO240,40:HPLOT 0,80 TO 240,80
1230 HPLOT 0,120 TO 240,120
1240 HPLOT 10,0 TO 10,159:HPLOT 60,0 TO 60,159
1250 HPLOT 70,0 TO 70,159:HPLOT 120,0 TO 120,159
1260 HPLOT 130,0 TO 130,159:HPLOT 180,0 TO 180,159
1270 HPLOT 190,0 TO 190,159:HPLOT 240,0 TO 240,159
1280 FOR I=0 TO 15:Y0=3+I*4
1290 HPLOT 245,Y0 TO 255,Y0:HPLOT 265,Y0 TO 275,Y0
1300 NEXT:FOR I=0 TO 15:Y0=83+I*4
1310 HPLOT 245,Y0 TO 255,Y0:HPLOT 265,Y0 TO 275,Y0
1320 NEXT
1330 SCALE=1
1340 FOR J=0 TO 15:Y0=INT(J/4)
1350 X0=J-Y0*4
1360 READNMS:FOR I=1 TO 3
1370 NM=ASC(MIDS(NMS,I,1))-31
1380 DRAW NM AT X0*60+2,Y0*40+I*8
1390 NEXTI:NEXTJ
1400 C=0:YN=.25:GOSUB1470:YN=.33:GOSUB1470
1403 YN=.47:GOSUB1470
1410 C=1:YN=.09:GOSUB1470:YN=.19:GOSUB1470
1411 YN=.26:GOSUB1470
1412 YN=.35:GOSUB1470:YN=.47:GOSUB1470

```

;DEF START OF CHAR SET
;NUMBER OF CHAR?

;END OF CHAR DEFS
;CHAR DEF ENDS WITH ZERO
;ENTER POS OF NEXT CHAR IN DIR

;LINES FOR AXES

;TITLE FOR EACH GRAPH

;GRADUATIONS FOR EACH GRAPH

```

1413 YN=.6:GOSUB1470:YN=.77:GOSUB1470
1430 FOR C=3 TO 4:YN=.13:GOSUB1470
1432 YN=.28:GOSUB1470:YN=.52:GOSUB1470
1433 YN=.79:GOSUB1470:NEXT
1440 FOR C=10 TO 14:YN=.11:GOSUB1470
1442 YN=.35:GOSUB1470:YN=.59:GOSUB1470
1443 YN=.83:GOSUB1470:NEXT
1445 FOR C=5 TO 8:FOR M=2 TO 8STEP2
1446 YN=M/10:GOSUB1470:NEXTM:NEXT
1460 GOTO1500
1470 Y0=1+INT(C/4)
1472 X0=(C+4-Y0*4)*60+8:XN=X0+4
1473 YN=(Y0-YN)*40:HPL0T X0,YN TO XN,YN:RETURN
1500 POKE 3073,00:POKE 3074,32
1510 POKE 3075,255:POKE 3076,63
1520 POKE 3077,00:POKE 3078,64
1530 POKE 3079,0:CALL3082
4900 END
4910 PRINTCHR$(4);"RUN H2 PC"
5000 DATA 70
6390 REM SPECIAL CHAR
6400 DATA 73,73,0
6420 DATA 73,73,0
6440 DATA 73,73,0
6460 DATA 73,73,0
6480 DATA 72,73,28,28,28,28,12,21
6490 DATA 30,30,118,101,12,149,2,0
6500 DATA 73,73,0
6520 DATA 73,73,0
6540 DATA 73,73,0
6560 DATA 72,9,28,36,100,141,146,18,0
6580 DATA 72,12,36,228,77,145,146,2,0
6600 DATA 8,12,12,28,28,77,241,22,14,21,0
6620 DATA 72,33,36,228,147,42,45,173,18,0
6640 DATA 73,12,100,145,2,0
6660 DATA 8,24,8,45,45,149,2,0
6680 DATA 72,105,17,0
6700 DATA 8,12,12,12,12,149,146,0
6710 REM NUMBERS 0-9
6720 DATA 72,45,12,36,228,63,23,54
6730 DATA 46,12,12,150,74,0
6740 DATA 72,37,36,36,23,173,146,78,0
6760 DATA 8,12,12,12,228,191,150,42,45,173,0
6780 DATA 8,112,45,12,228,43,228,27
6790 DATA 45,45,149,146,2,0
6800 DATA 72,33,60,39,12,12,149,42,149,2,0
6820 DATA 8,112,45,12,228,63,39,44
6830 DATA 45,173,146,18,0
6840 DATA 72,45,12,228,247,30,36,100
6850 DATA 45,141,146,18,0
6860 DATA 72,33,100,12,220,27,45,45
6870 DATA 149,146,2,0
6880 DATA 72,45,220,35,12,28,12,45
6890 DATA 21,222,173,182,1,0
6900 DATA 72,45,12,36,228,63,30,118,45,173,18,0
6910 REM SPECIAL CHAR
6920 DATA 8,72,4,24,8,77,146,18,0
6940 DATA 9,12,36,24,8,77,146,18,0
6960 DATA 72,9,28,28,12,12,141,146,2,0

```

;TRANSFER TO AUX MEMORY

; LOAD AND START MAIN PROGRAM

6980 DATA 8,40,45,37,24,216,43,45,173,146,2,0
7000 DATA 72,12,12,28,28,77,145,146,0
7020 DATA 72,33,32,12,12,28,63,30,150,146,73,9,0
7040 DATA 73,73,0
7050 REM ALPHABET A-J
7060 DATA 8,36,100,12,14,14,222,43,45,54,14,0
7080 DATA 8,36,36,44,45,21,30,63,150,45,12,172,18,0
7100 DATA 8,32,36,12,45,14,150,222,43,109,2,0
7120 DATA 72,9,63,39,36,36,45,141,54,182,1,0
7140 DATA 8,36,36,44,45,181,59,183,42,45,21,0
7160 DATA 8,36,36,44,45,181,59,183,82,73,0
7180 DATA 72,45,12,60,247,35,36,12
7190 DATA 45,21,142,146,0
7200 DATA 8,36,36,172,42,45,36,150,54,14,0
7220 DATA 72,37,36,228,45,150,18,141,0
7240 DATA 8,112,101,36,228,45,149,146,2,0
7250 REM ALPHABET K-T
7260 DATA 8,36,36,108,241,30,14,14,14,21,0
7280 DATA 8,36,36,172,146,42,45,21,0
7300 DATA 8,36,36,12,149,100,21,54,54,14,0
7320 DATA 8,36,36,172,14,14,12,36,54,54,118,0
7340 DATA 72,45,12,36,228,63,30,54,182,73,9,0
7360 DATA 8,36,36,44,45,21,246,63,150,73,1,0
7380 DATA 72,73,28,28,22,231,36,100,45,21,54,142,2,0
7400 DATA 8,36,36,44,45,21,246,27,45,118,14,0
7420 DATA 8,112,45,12,28,28,231,12,45,14,142,146,0
7440 DATA 72,33,36,228,43,45,173,146,18,0
7450 REM ALPHABET U-Z
7460 DATA 8,32,36,172,146,42,101,36,36,149,146,2,0
7480 DATA 72,225,28,36,108,9,54,246,86,1,0
7500 DATA 72,28,36,36,141,18,118,12,36,36,149,146,2,0
7520 DATA 8,100,12,28,28,77,241,22,14,118,0
7540 DATA 72,33,36,231,108,9,246,150,74,0
7560 DATA 8,100,12,12,220,43,45,181,59,191,50,45,45,21,0
7570 REM SPECIAL CHAR
7580 DATA 73,73,0
7600 DATA 73,73,0
7620 DATA 73,73,0
7640 DATA 73,73,0
7660 DATA 45,45,45,29,0
7680 DATA 73,36,36,36,172,146,146,9,0
7700 DATA 30,30,77,225,28,54,54,54,6,0
7720 DATA 73,73,0
7740 DATA 73,73,0
7760 DATA 73,73,0
7780 DATA -1
7990 REM A<->M MEM TRANSFER ML PROG
8000 DATA 173,1,12,133,60,173,2,12
8010 DATA 133,61,173,3,12,133,62,173
8020 DATA 4,12,133,63,173,5,12,133
8030 DATA 66,173,6,12,133,67,173,7
8040 DATA 12,201,1,240,4,56,76,52
8050 DATA 12,24,32,17,195,96
8200 DATA TH1,TH2,VR5,GP1,GP4,LSB,LST,TC1
8210 DATA TC2,N/U,GP2,GP3,GP6,GP7,GP5,N/U
10000 DATA -1

E.5: Data Transfer Subroutine

<u>Memory Location</u>	<u>Decimal Content</u>	<u>Hex Content</u>	<u>Hex Loc.</u>	<u>Assembly Language</u>
3082-84	173 01 12	AD 01 0C	C0A	LDA C01 ;3082-3111
3085-86	133 60	85 3C	C0D	STA 3C ;stores
3087-89	173 02 12	AD 02 0C	C0F	LDA C02 ;required
3090-91	133 61	85 3D	C12	STA 3D ;locations
3092-94	173 03 12	AD 03 0C	C14	LDA C03 ;for
3095-96	133 62	85 3E	C0D	STA 3E ;firmware.
3097-99	173 04 12	AD 04 0C	C0F	LDA C04
3100-01	133 63	85 3F	C12	STA 3F
3102-04	173 05 12	AD 05 0C	C1E	LDA C05
3105-06	133 66	85 42	C21	STA 42
3107-09	173 06 12	AD 06 0C	C23	LDA C06
3110-11	133 67	85 43	C26	STA 43
3112-14	173 07 12	AD 07 0C	C28	LDA C07 ;determine
3115-16	201 01	C9 01	C2B	CMP #1 ;direction
3117-18	240 04	F0 04	C2D	BEQ 04 ;of transfer
3119	56	38	C2F	SEC
3120-22	76 52 12	4C 34 0C	C30	JMP C34
3123	24	18	C33	CLC
3124-26	32 17 195	20 11 C3	C34	JSR C311 ;call
3127	96	60	C37	RTS ;firmware